Rivista + "Le grandi guide di ioProgrammo" n°6 a € 12,90 in più
VERSIONE PLUS RIVISTA+LIBRO+CD €9,90
VERSIONE STANDARD RIVISTA+CD €6,90

ioPROGRAMMO

PER ESPERTI E PRINCIPIANTI

Poste Italiane S.p.A Spedizione in A.P. • D.L. 353/2003 (conv.in L.27/02/2004 n.46) art.1 comma 2 DCB ROMA Periodicità mensile • SETTEMBRE 2006 • ANNO X, N.9 (106)

MICROSOFT POWERSHELL

QUASI COME LA BASH DI UNIX. ECCO COME SI USA IL LINGUAGGIO DI SCRIPTING DI WINDOWS

FATTI LA TUA SMART CARD

Dalla carta di credito alla scheda del telefono, ecco come si programmano i chip del futuro

TEORIA L'architettura e gli standard di riferimento

STRUMENTI Il platform invoke: la porta d'accesso verso i circuiti

PRATICA Crea un'applicazione che funziona solo se la tua carta è inserita nel lettore

CODICE Gli esempi in C# e Visual Basic.NET

JAVA

JAVA CONTROLLA MS OFFICE

Da Word ad Excel: ecco le tecniche per manovrarli direttamente dal tuo codice

SEMPRE PIÙ VELOCI CON I PROFILER

Impara a usare Jrat, per sapere esattamente perché e dove il tuo software rallenta

SNMP: LO 007 DEL SISTEMA

Crea un'applicazione che ti avverte quando qualche parametro è fuori controllo!

.NET

METTI UN FTP NEL BROWSER

Un completo client che gira dentro Explorer. Utile per consentire l'accesso da remoto...

PROGRAMMA SUBITO CON ASP.NET

PROGRAMMA IN XML OTTIENI UN FLASH!

Incredibile! Il nuovo Adobe Flex 2.0 produce applicazioni multipiattaforma, veloci e belle da vedere. Quasi Java ma più facile da usare...

USA L'HARD DISK COME DATABASE

Crea un Custom Provider ed esegui Query per cercare informazioni sull'HD

I NUOVI FRAMEWORK

ARRIVA IL GOOGLE WEB TOOLKIT

Vuoi programmare in Ajax? Ecco lo strumento pensato dal Re della rete per sviluppare le applicazioni per il Web! Ti insegniamo come usarlo

IL DESKTOP VA SU INTERNET

Usa le Web Parts per poggiarci sopra gli oggetti e gestirli con il mouse

MOBILE

PALMARI E DATABASE

Hai i dati su un server centrale? Ecco come aggiornarli senza generare conflitti!

SISTEMA

DEBUG SOTTO CONTROLLO

Come personalizzare Visual Studio per ottenere report avanzati

PATTERN

A LEZIONE DI SINGLETON

La tecnica per non avere oggetti duplicati e migliorare l'efficienza del software

SICUREZZA Privacy mai più violata grazie alla crittografia asimmetrica. Tutti gli algoritmi visti al microscopio

EDIZIONI MASTER
www.edmaster.it

60106
9 771128 594641

Anno X - N.ro 9 (106) - Settembre 2006 - Periodicità Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale: Massimo Sesti
Direttore Responsabile: Massimo Sesti
Responsabile Editoriale: Gianmarco Bruni
Redazione: Fabio Farnesi
Collaboratori: C. Bellucci, L. Buono, D. De Michelis, F. Grimaldi, M. Scala, A. Pelleriti, M. Locuratolo, L. Corias

Segreteria di Redazione: Veronica Longo

Realizzazione grafica: Cromatika S.r.l.
Art Director: Paolo Cristiano
Responsabile grafico di progetto Salvatore Vuono
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Francesco Cospite

Realizzazione Multimediale: SET S.r.l.
Realizzazione CD-Rom: Paolo Iacona

Pubblicità: Master Advertising s.r.l.
Via C. Correnti, 1 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite: Daisy Zonato

Editore: Edizioni Master S.p.a.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Presidente e Amministratore Delegato: Massimo Sesti
Direttore Generale: Massimo Rizzo

ABBONAMENTO E ARRETRATI

ITALIA: Abbonamento Annuale: ioProgrammo (11 numeri) €59,90 sconto 20% sul prezzo di copertina di €75,90 · ioProgrammo con Libro (11 numeri) €75,90 sconto 30% sul prezzo di copertina di €108,90 Offerte valide fino al 30/09/06 Costo arretrati (a copia): il doppio del prezzo di copertina + € 5.32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.

La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via C. Correnti, 1 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- cc/p n.16821878 o vaglia postale (inviando copia della ricevuta del versamento insieme alla richiesta);
- assegno bancario non trasferibile (da inviarsi in busta chiusa insieme alla richiesta);
- carta di credito, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- bonifico bancario intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.

Sostituzioni: qualora nei prodotti fossero rinvenuti difetti o imperfezioni che ne limitassero la fruizione da parte dell'utente, è prevista la sostituzione gratuita, previo invio del materiale difettato.
La sostituzione sarà effettuata se il problema sarà riscontrato e segnalato entro e non oltre 10 giorni dalla data effettiva di acquisto in edicola e nei punti vendita autorizzati, facendo fede il timbro postale di restituzione del materiale.
Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master - Servizio Clienti - Via C. Correnti, 1 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel. 02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Arti Grafiche Boccia S.p.a. Via Tiberio Felice, 7 Salerno
Stampa CD-Rom: Neotek S.r.l. - C.da Imperatore - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Agosto 2006

Audio Video Foto Bild, .A-Team, Calcio & Scommesse, Colombo, Computer Bild Italia, Computer Games Gold, Digital Japan Magazine, Digital Music, Distretto di polizia in DVD, DVD Magazine, Family DVD Games, Filmteca in DVD, Giochi e Programmi per il tuo telefonino, Go!OnLine Internet Magazine, Guide di Win Magazine, Guide Strategiche di Win Magazine giochi, Home Entertainment, Horror mania, I Corsi di Win Magazine, I Fantastici CD-Rom, I film di idea web,I Filmissimi in DVD, I Libri di Quale Computer, I Mitici all'Italiana, Idea Web, InDVD, IoProgrammo, Japan Cartoon, La mia Barca, La mia Videoteca, Le Femme Fatale del Cinema, Le Grandi Guide di Io Programmo, Linux Magazine, Magnum PI, Miami Vice in DVD, MPC, Nightmare, Office Magazine, Play Generation, Popeye, PC Junior, PC VideoGuide, Quale Computer, Softline Software World, Supercar in dvd, Thriller Mania, Win Junior, Win Magazine Giochi, Win Magazine, Le Collection.

FRAMEWORK EVOLUTION

Così la vera rivoluzione del momento sembra essere l'annuncio del rilascio di una versione Beta dello Zend Framework. Ora non è che si tratti di un annuncio sostanziale, esistono già tonnellate di classi che mirano ad estendere le già elevate potenzialità di PHP. Si tratta però di un annucnio importante perché a farlo è ZEND, principale promotrice dello sviluppo del linguaggio.
La novità è soprattutto di marketing. Ricalcando un po' le orme del .NET framework almeno nel nome, si vuole lasciare intendere che quello che esiste sopra il livello delle dll di .NET e cioè una serie di classi omogenee tese a rendere semplice la vita al programmatore, esiste in realtà anche per PHP e da molto più tempo. Ora non è che fra i due linguaggi ci sia concorrenza. PHP rimane il leader indiscusso delle Web application di medio livello su Internet. E' dal punto di vista delle Business application che però si combatte la vera battaglia. In questo settore PHP non ha la stessa diffusione di .NET o di JSP e tuttavia è proprio qui che risiedono i fondi che servono per mantenere in attivo società come ZEND e che devono essere utilizzati per finanziare lo sviluppo di PHP. Dal punto di vista strettamente tecnico, lo ZEND framework si configura come un insieme di classi realmente ben strutturato. Sono sufficienti poche righe di codice per programmare un Web Service o per creare un report in formato PDF. Zend ha decisamente fatto un buon lavoro. Cosa manca dunque a PHP? Probabilmente soltanto un editor visuale del calibro di Visual Studio. Ma in realtà la forza di PHP sta probabilmente nella flessibilità, per cui legarlo strettamente a un tipo di editor sarebbe probabilmente un boomerang.
Dove voglio arrivare con questo mio ragionamento? Voglio semplicemente mostrare ai programmatori quanto possa essere ampia la scelta degli strumenti disponibili sul mercato e che tendono a migliorare la produttività del nostro lavoro. Molto probabilmente la questione è scegliere sempre quello giusto per le nostre esigenze. Ormai il mercato è talmente vasto che affidarsi a un framework come quello di Zend significa non dover riscrivere da zero i l proprio codice. L'importante è avere la forza di volontà di imparare un nuovo framework!

Fabio Farnesi ffarnesi@edmaster.it

All'inizio di ogni articolo, troverete un simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.

FATTI LA TUA SMART CARD

Dalla carta di credito alla scheda del telefono, ecco come si programmano i chip del futuro

- ✔ **TEORIA:** L'architettura e gli standard di riferimento
- ✔ **CODICE:** gli esempi in C# e Visual Basic.NET
- ✔ **PRATICA:** Crea un'applicazione che funziona solo se la tua carta è inserita nel lettore
- ✔ **STRUMENTI:** Il platform invoke e la porta d'accesso verso i circuiti

pag. 14

PROGRAMMA IN XML OTTIENI UN FLASH!

Il nuovo Adobe Flex 2.0 produce applicazioni multipiattaforma, veloci e belle da vedere. Quasi Java ma più facile da usare

IOPROGRAMMO WEB

Una guida passo passo alla nuova versione del prodotto di Adobe dedicato agli sviluppatori. Vedremo come in un modo rapido e visuale si possono sviluppare interfacce che funzionano sia per Web sia standalone e per tutte le piattaforme

Al gran numero di Framework per sviluppare applicazioni Web in tecnologia Ajax non poteva mancare un toolkit sviluppato dal gigante di Internet. Il Google Web Toolkit si presenta con qualche carta in più. Vediamo quale

Una delle novità più interessanti del Framework .Net 2.0 è sicuramente quella delle "Web parts". Si tratta di un metodo che consente ad un utente di costruire dinamicamente una pagina Web adattandola alle proprie preferenze...

DATA BASE

Indagheremo più a fondo nelle potenzialità del framework Hibernate. Presenteremo un plugin che ci permetterà di sviluppare Gui all'interno di Eclipse. Impareremo come i database possano essere trattati come oggetti

Implementiamo un componente intermedio che consente di accedere a qualunque tipo di dato utilizzando gli stessi metodi ed interfacce e variando solo pochi parametri

NETWORKING

Metti un FTP nel tuo Browser

Creiamo un client FTP che vive all'interno di Explorer. Utilissimo per uploadare file in remoto senza scomodare programmi di terze parti

MOBILE

Se abbiamo una rete di palmari, i dati presenti in un dispositivo devono sicuramente essere sincronizzati con un database centrale. Scopriamo come gestire upload multipli evitando eventuali conflitti

SISTEMA

I profiler sono strumenti molto interessanti che consentono di ottenere una misura delle prestazioni precisa mentre il sofware è in esecuzione. Sfruttando queste tecniche è possibile ottenere miglioramenti significativi...

Siamo abituati a pensare a Java come un mondo completamente staccato da quello dei software che girano sui sistemi microsoft. Tuttavia esistono soluzioni che possono rendere i due mondi molto vicini...

Non tutti sanno che visual studio 2005 offre possibilità di personalizzazione praticamente infinite. in questo articolo illustreremo come migliorare il debug di strutture proprietarie , utilizzando controlli specifici

RUBRICHE

Il software in allegato alla rivista

Il contenuto del libro in allegato alla rivista

Le più importanti novità del mondo della programmazione

I contenuti del CD allegato ad ioProgrammo.

In questo articolo illustreremo alcune caratteristiche del nuovo framework 2.0 che consentono di accedere rapidamente a informazioni di sistema o altri percorsi di progetto, senza troppe complicazioni.

NETWORKING

SNMP: sistema sotto controllo

Java dispone di alcune classi che consentono di interfacciare le nostre applicazioni con SNMP

Gli utenti Unix sono abituati a lavorare a riga di comando. Dopo tanti anni anche Microsoft ha deciso di fornire strumenti avanzati di automazione per la gestione del sistema basati su Shell. Vediamo quali

Non molti conoscono questa tecnica di programmazione. Tuttavia si tratta di un tassello fondamentale per non inciampare in oggetti duplicati e strutture poco efficienti. Impariamo cosa e come funziona

QUALCHE CONSIGLIO UTILE

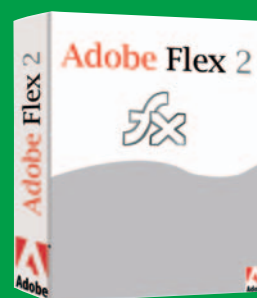
I nostri articoli si sforzano di essere comprensibili a tutti coloro che ci seguono. Nel caso in cui abbiate difficoltà nel comprendere esattamente il senso di una spiegazione tecnica, è utile aprire il codice allegato all'articolo e seguire passo passo quanto viene spiegato tenendo d'occhio l'intero progetto. Spesso per questioni di spazio non possiamo inserire il codice nella sua interezza nel corpo dell'articolo. Ci limitiamo a inserire le parti necessarie alla stretta comprensione della tecnica.

http://forum.ioprogrammo.it

RIVISTA + CD-ROM in edicola

FLEX 2.0

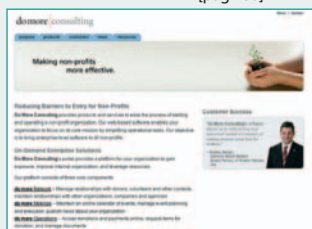
Il nuovo Framework di Adobe

Straordinario! Non ci sono parole per descrivere questo nuovo prodotto della linea Adobe che in un sol colpo fa piazza pulita di un vecchio modo di pensare alle applicazioni per inventarne uno tutto nuovo, semplice ed estremamente potente. Chi ha letto questo numero di ioProgrammo sa già come funziona Flex 2.0. Si scrive un programma utilizzando un dialetto di XML, lo si da in pasto al compilatore e si ottiene in uscita un file SWF ovvero utilizzabile dal noto player Flash di macromedia. Quali sono i vantaggi? Ce ne sono diversi, i più facili da intuire sono la completa indipendenza dal sistema operativo senza per questo doversi affidare alla macchinosità di java. La possibilità che la stessa applicazione possa girare praticamente senza modifiche sia in modo standalone che nel browser. La possibilità di sviluppare interfacce dall'aspetto omogeneo e molto altro...

Prodotti del mese

Ajax Pro Starter Kit

Per iniziare facilmente con Visual Studio

Da qualche tempo non si fa altro che parlare di Ajax. Si tratta della nuova tecnologia che consente di realizzare pagine web dinamiche che non necessitano del reload per aggiornare i dati. Come potete intuire si tratta di un'innovazione senza precedenti che consente di realizzare applicazioni Web con comportamenti molti simili alle normali applicazioni Desktop. Visual Studio 2005 dispone della tecnologia degli Starter Kit, che consente di iniziare un progetto partendo da una sorta di template che ne facilità l'avvio. Il tool che vi presentiamo è appunto lo starter kit che consente di iniziare a sviluppare un progetto Web in ASP.NET ed in tecnologia .NET. Si tratta senza dubbio di un valido aiuto per sviluppare le proprie Web Application utilizzando questa tecnica

[pag.106]

Atlas Control Toolkit

I migliori componenti per lo sviluppo Web

Nasce da una collaborazione fra Microsoft e i creatori originali, questo interessante pacchetto che si propone di fornire al programmatore una serie di componenti "client" adatti a favorire lo sviluppo di Web Application. All'interno del pacchetto si trova una ricca collezione di esempi che fanno uso di una quindicina di nuovi controlli, infine c'è un SDK che consente di estendere ulteriormente Atlas che in questo modo si configura come un vero e proprio Framework per lo sviluppo. La quantità e la qualità dei controlli è tale che utilizzando questo pacchetto si ottengono vantaggi sia in relazione alla produttività sia in relazione all'eficienza

[pag.106]

Hibernate 3.2.0

Trasforma i da dati da SQL ad Oggetti

Ormai lo sappiamo abbiamo letto tutti gli articoli di ioProgrammo su Hibernate, pensare in termini di query e dati SQL non è più conveniente! Disponiamo di oggetti e di classi, non possiamo continuare a lavorare in modo disomogeneo trattando i dati SQL in maniera separata dal resto del flusso del programma. E per mappare i dati da SQL ad oggetti abbiamo bisogno di un tool. Il primo ed ancora adesso Leader di questo settore è Hibernate, attraverso il quale riusciamo ad ottenere comodamente i risultati voluti. Il meccanismo è semplice, si scrivono alcuni file XML che rappresentano la descrizione del nostro DB, si danno in pasto ad Hibernate che provvede a creare una serie di classi e di oggetti utili a gestire il database, liberandoci dal vincolo di doverli progettare a mano

[pag.107]

PHP 5.1.4

Il linguaggio più utilizzato per la programnazione di internet

Se seguite ioProgrammo o più semplicemente siete dei programmatori Web, o ancora molto più semplicemente navigate su Internet, non potete non sapere che cosa è PHP. Si tratta del linguaggio con il quale sono sviluppate la maggior parte delle applicazioni internet esistenti. Quasi tutto il software per il web si regge su PHP. La curva di apprendimento è bassissima, le funzionalità esposte elevatissime, certamente se avete intenzione di sviluppare per il web non potrete fare a meno di provare anche questo linguaggio come base per le vostre applicazione. Attualmente la versione 5.1.4 espone un modello ad oggetti piuttosto completo che rende PHP un linguaggio moderno dalla curva di apprendimento molto rapida e persino didattico

[pag.107]

I contenuti del libro

Lavorare con C++

C++ è il linguaggio più amato da chi programma a basso livello. Per certi versi può considerarsi il padre di tutti i linguaggi. Sicuramente quello a cui tutti si sono ispirati per creare nuovi paradigmi di programmazione. E' sicuramente un linguaggio multipiattaforma. Chi lo padroneggia può ritenere a ragione di conoscere i sistemi in modo profondo. In questo libro "Roberto Allegra" ci mostra i concetti avanzati del linguaggio. Si tratta di un viaggio affascinante, condotto spesso al bordo della conoscenza profonda delle macchine e del funzionamento dei sistemi operativi. Un libro da leggere, non solo e soltanto per impadronirvi delle tecniche necessarie a programmare il C++, ma anche per comprendere a fondo certi meccanismi dell'informatiche che a molti risultano oscuri

LE STRUTTURE E LE FUNZIONI AVANZATE. PER SFRUTTARE AL MASSIMO IL PADRE DI TUTTI I LINGUAGGI

- **Gestione dinamica degli oggetti**
- **Utilizzare le eccezioni**
- **Le librerie e i canali standard**
- **Stringhe e canali**

GLI ALLEGATI DI IOPROGRAMMO

▼ Web Development

Sviluppare Web Application professionali, in grado di soddisfare le esigenze sempre più complesse di un mercato in continua evoluzione implica un elevato grado di attenzione da dedicare ad ogni singolo particolare di un'applicazione. Non è sufficiente sviluppare soluzioni che risolvano un problema. E' invece necessario sviluppare software che sia stabile sicuro, efficace e veloce. In questo contesto ASP.NET si propone come un framework che dota lo sviluppatore di strumenti avanzati, che inglobano per loro natura le caratteristiche richieste. Conoscere questo genere di strumenti significa compiere il necessario salto di qualità per essere competitivi sul mercato dello sviluppo. In questi tre Webcast MSDN vengono affrontate due delle principali tecnologie coinvolte nello sviluppo di Web Application professionali. Nel primo caso si parla di interazione con i database, là dove un controllo minuzioso su come i dati vengono manipolati garantisce una maggiore efficacia all'applicazione. Nel secondo caso si parla di deployment di una soluzione, operazione che a maggior ragione per .NET si presenta come molto delicata. Prima perché .NET porta in sé un'innata propensione alla sicurezza, che però deve essere supportata da un deployment efficace, secondo perché il deployment può avvenire in diverse situazioni non sempre omogenee.

I VIDEOCORSI PER PROGRAMMARE BENE

Imperdibili! Per imparare a programmare applicazioni Web Dinamiche con supporto ai database

msdn WEBCAST

- **ASP.NET 2.0 DataBinding**
- **ASP.NET 2.0 DataBinding avanzato**
- **ASP.NET 2.0 Deployment di una soluzione**

INFORMAZIONI SU MSDN WEBCAST
http://www.microsoft.it/msdn/webcast_msdn
http://forum.ioprogrammo.it

FAQ

Cosa sono i Webcast MSDN?

MSDN propone agli sviluppatori una serie di eventi gratuiti online e interattivi, che approfondiscono le principali tematiche relative allo sviluppo di applicazioni su tecnologia Microsoft. Questa serie di "corsi" sono noti con il nome di Webcast MSDN.

Come è composto tipicamente un Webcast?

Normalmente viene presentata una serie di slide commentate da un esperto di tecnologie Microsoft. A supporto di queste presentazioni spesso vengono realizzate delle demo in presa diretta che mostrano dal vivo come usare le tecnologie oggetto del Webcast

Come mai nei webcast allegati alla rivista si parla di chat e di altri strumenti interattivi?

La natura dei Webcast è quella di essere seguiti online e in tempo reale. Durante queste presentazioni in diretta vengono utilizzati strumenti molto simili a quelli della formazione a distanza. E' possibile porre domande in presa diretta al relatore oppure partecipare ai sondaggi che vengono proposti. In questo modo gli sviluppatori possono aggiornarsi e approfondire i temi di loro interesse con maggiore efficacia. I Webcast riprodotti nel CD di ioProgrammo, pur non perdendo nessun contenuto informativo, per la natura asincrona del supporto non possono godere dell'interazione diretta con il relatore.

Come mai trovo i Webcast su ioProgrammo

Come sempre ioProgrammo cerca di fornire un servizio ai programmatori italiani. Abbiamo pensato che poter usufruire dei Webcast MSDN direttamente da un CD rappresentasse un ottimo modo di formarsi comodamente a casa e nei tempi desiderati. Lo scopo tanto di ioProgrammo, quanto di Microsoft è infatti quello di supportare la comunità degli sviluppatori italiani con la più ampia gamma di strumenti di formazione e aggiornamento.

Su ioProgrammo troverò tutti Webcast MSDN?

Ne troverai sicuramente una buona parte. Direttamente sul sito MSDN potrai consultare il calendario dei prossimi webcast a cui iscriverti per seguirli in diretta oppure consultare l'archivio delle registrazioni di quelli già realizzati. L'indirizzo per saperne di più è www.microsoft.it/msdn/webcast_msdn, segnalo nel tuo bookmark, non può mancare!

L'iniziativa sarà ripetuta sui prossimi numeri?

Sicuramente sì, alla prossima.

News

MICROSOFT REGALA VIRTUAL PC

La virtualizzazione è la vera novità degli ultimi anni per noi programmatori. Sia VMWare sia Virtual PC consentono di far girare più sistemi operativi in una sorta di SandBox. Questo ci permette di testare il software nel migliore dei modi su più sistemi e sotto diverse condizioni senza per questo dover continuamente formattare la macchina su cui eseguiamo lo sviluppo.
Recentemente VMWare ha annunciato il rilascio gratuito del proprio Player. Un passo ancora maggiore lo ha compiuto Microsoft annunciando di voler distribuire gratuitamente il proprio Virtual PC 2004. Questa mossa è sicuramente da intendersi come il tentativo di conquistare nuove fette di mercato in vista dell'arrivo di Virtual PC 2007. Non è chiaro se la nuova versione sarà anche essa gratuita. Sicuramente la volontà è quella di rivaleggiare ad armi pari con il concorrente VMWare che fino ad ora è il leader di mercato. Vedremo come finirà questa entusiasmante sfida

SYMBIAN OS EVOLUZIONE 9.3

In un momento di forte espansione dei pocket PC dovuta principalmente alla buona riuscita di Windows Mobile 5.0, il rivale Symbian non poteva stare a guardare. Così è recente il rilascio della nuova versione di Symbian OS, la 9.3 che contiene alcune novità largamente attese. Finalmente è stato incluso nel sistema il supporto per il Wi-Fi. Sulla base di questa è stato poi aggiunto un supporto per il Voip, in questo modo gli utenti dovrebbero poter scegliere se telefonare in modo tradizionale oppure attraverso il Wireless. Novità molto importanti per noi sviluppatori è invece rappresentata dall'arrivo di alcuni strumenti di sviluppo basati sull'onnipresente Eclipse, e che dovrebbero garantirci una maggiore semplicità di sviluppo. Queste due importanti innovazioni dovrebbero fronteggiare la crescita di Windows Mobile 5.0

L'ACCESSIBILITÀ DELLA DISCORDIA

E' ancora la legge sull'accessibilità dei siti web a provocare un intenso dibattito nel parlamento come negli addetti ai lavori. La legge attuale è stata largamente disattesa, sia per motivi legati ad una certa "fumosità" in alcuni paragrafi della stessa, sia per la presenza di alcune "scappatoie" che hanno favorito i soliti furbi. Così Cesare Campa e Antonio Palmieri deputati di forza Italia si sono fatti carico di una nuova proposta elaborata ancora una volta con la collaborazione di IWA/HWG. Le questioni su cui si dibatte sono essenzialmente due. La prima riguarda l'obbligo del rispetto delle norme di accessibilità anche per i progetti nati come intranet. L'attuale legge infatti reca un buco normativo tale che molti progetti della P.A. nati all'interno delle strutture sono riusciti ad arrivare ai cittadini senza curarsi degli obblighi di accessibilità proprio perché l'attuale legge non prevede il rispetto della normativa per i progetti nati come intranet. In questo modo molte P.A. hanno potuto far nascere i progetti come interni alla propria struttura, esportandoli poi all'esterno senza per questo doversi sottoporre al controllo del CNIPA.

GOOGLE VS PAYPAL

Tutti conoscono PayPal, il sistema di pagamento elettronico più diffuso in america e che sta lentamente guadagnando posizioni anche in Italia. Il funzionamento è semplice. Ci si iscrive a Paypal, e si lasciano i dati della propria carta di credito. A questo punto non è più necessario fornire il proprio numero di carta di credito a nessun esercente. Sarà sufficiente fornire il proprio identificativo di accesso su paypal per dare il via alla transazione. Sarà paypal stesso a fare da tramite fra gli istituti di credito. Questo metodo si è rivelato un enorme successo ed in molti hanno cercato di imitarlo senza tuttavia riuscirvi pienamente. E' adesso il turno di Google che ha appena lanciato sul mercato un sistema che prende il nome di Chekout e che utilizza lo stesso identico sistema di paypal. Ovviamente staremo a vedere chi la spunterà e quali sono i vantaggi nell'usare l'uno e l'altro metodo. Attualmente checkout non sembra avere la stessa penetrazione di mercato che ha avuto paypal al suo lancio. Tuttavia trattandosi di Google possiamo scommettere sulla validità del servizio. Si registrano già le prime reazioni di Ebay che è anche proprietaria di Paypal e che ovviamente si è dichiarata molto ostile all'arrivo di questo nuovo soggetto finanziario

Il secondo punto di discussione riguarda proprio il CNIPA. Attualmente è previsto che questo ente provveda a monitorare solo le amministrazioni centrali. Il nuovo progetto prevede che il controllo passi al Corecom (comitato regionale per le comunicazioni) che dovrebbe poter monitorare anche regioni, province, ed enti locali. La legge è di quelle importanti, perché una sua corretta applicazione garantirebbe anche ai cittadini diversamente abili un corretto accesso alle informazioni e ai servizi contenuti in rete. Al di la del merito delle proposte che sono state effettuate, noi pensiamo che sia necessaria una sensibilizzazione su grande scala in relazione all'argomento, proprio perché le software house devono essere le prime responsabili nell'attuazione corretta dei regolamenti e soprattutto nell'applicazione delle regole di buon senso che renderebbero fruibili i contenuti OnLine al maggior numero di persone possibili

ARRIVA LO ZEND FRAMEWORK

Che a far la fortuna di PHP, fra le altre cose, sia la sua estrema scalabilità è un fatto noto. Chi usa PHP può scegliere di programmare in modo Object Oriented, in modo procedurale e persino in modo Flat. A tutto ciò si aggiunge un cospicuo numero di progetti e classi esterne che ne possono estendere le possibilità. E' di questi giorni l'annuncio del rilascio di una prerelease dello "Zend Framework", in particolare della sua versione 0.1.4. Si tratta di un annuncio di particolare rilievo se si pensa che Zend è l'azienda principale promotrice dello sviluppo di PHP. In teoria lo Zend framework non si distanzia di molto a livello concettuale dai tool già esistenti, ad esempio PEAR. Si tratta cioè di una serie di classi che lavorano sotto un unico "cappello". All'atto pratico però lo Zend Framework si è rivelato particolarmente potente, ben costruito e affidabile. Sono sufficienti un paio di righe di codice per sviluppare un Web Services, non molte di più per creare un report in formato PDF. Le classi presenti nello Zend Framework coprono sostanzialmente tutti gli aspetti nodali dello sviluppo software in ambiente Web. Si tratta di un passo importante perché adesso gli sviluppatori hanno a disposizione uno strumento di alto livello per programmare le proprie applicazioni. Chi vuole potrà sempre cominciare ad apprendere il linguaggio e iniziare a scrivere tutto il codice partendo dalle classi base, gli altri possono imparare direttamente a usare il FrameWork e utilizzare le sue peculiarità per sviluppare. In tutti i casi ancora una volta viene premiata l'eccezionale scalabilità di questo linguaggio

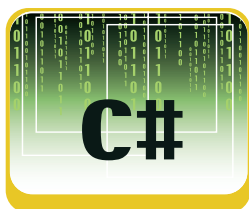
WINFS ADDIO!

Il nuovo rivoluzionario File System che Microsoft aveva annunciato come punto di forza del prossimo futuro Windows Vista non ce l'ha fatta a vedere la luce. Il progetto WinFS è stato annullato e non sarà più quest'ultimo a sostenere il File System di Vista. In realtà Microsoft non sembra avervi rinunciato totalmente. Secondo il big di Redmond, WinFS è stato semplicemente "smembrato" e buona parte delle tecnologie che lo sostenevano saranno integrate in altri progetti. In pratica WinFS continuerà a vivere, ma non come un progetto a se stante, quanto come una serie di "pillole" integrate in altre tecnologie. Questo almeno è quanto dichiarato da Microsoft. Diversa è l'opinione degli avversari dell'azienda di Bill Gates. Sono in molti infatti a sostenere che il progetto è stato abbandonato poiché diventato talmente complesso da non essere più gestibile. E' il classico cane che si morde la coda, ovvero un progetto talmente tecnologicamente avanzato che la sua complessità lo rende inaffidabile e instabile. Nel frattempo non ci resta che aspettare Vista per capire se il lavoro fin qui svolto ha realmente dato i risultati sperati

MICROSOFT ABBRACCIA L'OPENDOCUMENT

Con una mossa a sorpresa il colosso di Redmond ha annunciato di stare lavorando ad un progetto OpenSource che favorisca l'interoperabilità fra l'ODF e l'OPENXML.
Fino a qualche settimana fa il rifiuto di MS nei confronti di Open Document era netto. Secondo l'azienda di Bill Gates, l'interesse dei propri clienti nell'adottare l'ODF era basso. Oggi la posizione di Microsoft è radicalmente cambiata.
Con questo nuovo tool si intende fornire ai consumatori uno strumento in grado di convertire i documenti da un formato all'altro in modo estremamente semplice, dalla linea di comando o dall'interfaccia di Office.
Microsoft ha dichiarato di volere, in questo modo, sostenere fattivamente la tanto decantata interoperabilità. Tuttavia, questa operazione, sembra essersi resa necessaria anche a causa delle dichiarazioni del governo del Massachusetts che ha dichiarato di voler adottare proprio ODF come formato standard per i documenti della pubblica amministrazione. Il nuovo tool sarà compreso in Office 2007

PROGRAMMARE LE SMART CARD

POSSONO CONTENERE DATI. UTILIZZANO ALGORITIMI DI CRITTOGRAFIA MOLTO SICURI PER L'AUTENTICAZIONE. GIÀ OGGI SONO UTILIZZATE PER LA FIRMA DIGITALE. IN QUESTO ARTICOLO NE FAREMO UNA PANORAMICA COMPLETA...

Lo scopo di questo articolo è il seguente: impedire l'accesso ad un'applicazione a meno delle seguenti condizioni:

- un lettore di smart card sia connesso al computer
- l'utente che desidera accedere all'applicazione possieda una smart card e ne conosca il codice pin
- Se una volta immessa la Smart Card all'interno del lettore, l'utente è in grado di inserire il codice d'accesso corretto, allora avrà accesso all'applicazione, altrimenti no.

Tanto per fare un esempio tipico, è la classica situazione davanti alla quale ci si ritrova quando si preleva del denaro al bancomat. Inseriamo la nostra Smart Card (tessera bancomat) all'interno del lettore, se siamo in grado di verificarne il codice Pin saremo autorizzati a compiere le operazioni, altrimenti no. Questo tipo di protezione è largamente utilizzata in una serie di contesti. Abbiamo fatto l'esempio del bancomat, ma considerate anche il caso più semplice di un impiegato che potrà accedere al suo terminale solo previa autenticazione con Smart Card. E ancora quello della cassiera di un supermercato che potrà far funzionare la cassa solo se autenticata dalla sua Smart Card. In uno scenario più futuristico, su una Smart Card si possono conservare dati importanti, ad esempio i propri dati personali. In un futuro potrebbe essere sostitutiva della patente e già le camere di commercio italiane fanno uso di una Smart Card per concedere l'accesso ai propri servizi e per conservare la famosa "firma digitale".

Impegno

Tempo di realizzazione

COSA SERVE PER INIZIARE?

Avremo bisogno di un lettore di Smart Card e di qualche Smart Card di prova. Per il nostro

Fig. 1: ***Il lettore Omnikey Cardman nelle versioni USB e PCMCIA utilizzato per testare il presente articolo. Sulla sinistra due Smart Cartd infineon ancora vuote***

articolo abbiamo usato un lettore Omnikey CardMan 4040 PCMCIA acquistato presso Multimedia It – http://www.multimediait.it – e del costo di 69,00 . In alternativa sempre presso Multimedia It è possibile acquistare il CardMan 3121 che dispone di interfaccia USB e ha un costo di 25,00 . In realtà tutti i lettori di Smart Card dovrebbero funzionare con la tecnica proposta in questo articolo. Ma poiché come vedremo fra i vari lettori esiste qualche differenza, pur rimanendo la tecnica universalmente valida, potrebbe essere necessario riferirsi alla documentazione ufficiale del lettore utilizzato per avere piena padronanza dei codici da utilizzare. Per quanto riguarda il costo delle Smart Card, sempre presso Multimedia It è possibile acquistarle ad un prezzo che varia da 18.00 ai 25.00 a secondo del quantitativo acquistato.

LETTORI E SMART CARD

Il lettore di Smart Card è semplicemente un apparecchio che consente di accedere in lettura/scrittura alla Smart Card. Viceversa sulla Smart Card è presente un vero e proprio sistema operativo. Nel nostro caso abbiamo usato delle Smart Card prodotte da Infineon ed

equipaggiate con un sistema operativo Sicrypt conforme alle specifiche ISO 7816-4 e PKCS #15, vedremo a breve cosa significano queste sigle. Tutte le informazioni sul Sycript sono comunque disponibili sul sito http://www.infineon.com.
Un sistema operativo per Smart Card assolve a due funzioni particolari: gestire un file system con una struttura conforme alle specifiche di cui sopra, esporre dei comandi anche essi conformi alle ISO 7816-4.
Ad un livello molto "superficiale", possiamo dire che solo l'amministratore di sistema della Smart Card è in grado di scrivere su di essa. Pertanto prima di poter accedere alla smart card in modalità amministrativa, sarà necessario autenticarsi. Appare ovvio che la Smart Card deve mantenere al suo interno, in un'area protetta, la password dell'amministratore. Un client dall'esterno dovrà mandare al sistema operativo della Smart Card un "comando" che ha come parametro la password d'amministrazione. Il comando sarà uno di quelli esposti dal sistema operativo contenuto dalla Smart Card e sarà eseguito su di essa, verificherà che la password inviata come parametro sia identica a quella contenuta nel file system e se l'autenticazione risulta corretta restituirà un Ok e garantirà l'accesso a livello amministrativo. Riassumendo, per poter scrivere sulla Smart Card dovremo almeno compiere le seguenti operazioni

- Stabilire una connessione con la Smart Card
- Inviargli il comando: "Autenticami con Password"
- Aspettare il codice di ritorno

Il comando viene inviato al sistema operativo tramite una periferica esterna, ma è sempre il sistema presente sulla Smart Card ad eseguirlo. Un po' come succede tra la normale tastiera e il nostro computer. Il comando viene inviato dalla tastiera e dato in pasto al computer che lo esegue. Nel nostro caso, invieremo il comando da un software client presente sul computer, il lettore di Smart Card sarà la nostra interfaccia verso la Smart Card, il sistema operativo presente sulla Smart Card eseguirà il comando.

DISTRICARSI FRA LE SPECIFICHE

Abbiamo già detto che il sistema operativo Sicrypt di Infineon è conforme agli standard ISO 7816-4 e PKCS #15 ma cosa vuol dire esattamente questo? Le ISO7816 definiscono lo standard che deve essere adottato dalle Smart Card per poter essere compatibili con i lettori e con i software disponibili, descrive in sostanza: l' Identification cards – Integrated circuits cards with contacts. Ovvero l'insieme delle parti elettriche, le strutture fisiche e logiche che devono essere rispettate dai produttori di Smart Card. In particolare la struttura della ISO7816 si compone di 10 parti:

1. Caratteristiche fisiche
2. Dimensioni e posizione dei contatti;
3. Protocolli di trasmissione e segnali elettrici;
4. Comandi, messaggi e risposte tra smart card e lettore;
5. Regole di registrazione e identificazione delle applicazioni installate su Smart Card
6. Regole e formati dei dati precostituiti su una Smart Card e di quelli scambiati con il lettore.
7. Definisce la struttura logia di un DB per Smart Card e relativo SCQL, nonché i suoi comandi.
8. Comandi e protocolli di sicurezza.
9. Descrizione ciclo di vita di una smart card, dei suoi attributi di sicurezza.
10. Definisce la modalità di RESET (Answer to Reset) e i segnali elettrici.

Il sistema operativo Sicrypt è compatibile alla ISO 7816-4 ovvero alla quarta sezione delle specifiche in questione. Implementa cioè un set standard di comandi disponibili che resti-

Fig. 2: Architettura di un sistema di Smart Card

tuiscono messaggi anche essi aderenti allo standard.
Le intere specifiche ISO 7816-4 sono acquistabili all'indirizzo http://www.iso.org ad un costo di circa 120 . Per il nostro articolo, che ha scopo didattico e non necessita della conoscenza delle intere ISO, ci siamo avvalsi di una pagina informativa relativa ai comandi esposti dal sistema Sycript disponibile all'indirizzo http://www.infineon.com/cgi-bin/ifx/portal/ep/channelView.do?channelId=-75271&channelPage=%2Fep%2Fchannel%2FproductCategories.jsp&pageTypeId=17224

NOTE

ESEMPI E RIFERIMENTI

Per gli esempi prodotti in questo articolo ci siamo avvalsi di alcuni progetti OpenSource facilmente reperibili all'indirizzo http://www.gotdotnet.com Le tecniche qui descritte si sono basate proprio sullo studio di questi esempi

LO STANDARD PC/SC

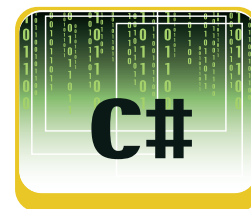
Nonostante che le ISO 7816 rappresentino la base per poter favorire l'interoperabilità tra le Smart Card, risultano troppo specifiche o troppo generiche in alcuni settori, per cui non sono state giudicate sufficienti a garantire la necessari uniformità fra le applicazioni. Per questo motivo nel 1996, Microsoft, Hewlett Packard, Shlumberger e Gemplus hanno dato vita a un gruppo di lavoro da cui è nato lo standard PC/SC. Tale sistema, non solo risolve il problema dell'interoperabilità tra smart card e lettore di diversa fabbricazione, ma fa si che lo sviluppatore si concentri sulla programmazione e non sull'architettura di sistema. Per capire meglio la funzionalità dello standard PC/SC, diamo uno sguardo alla sua struttura. PC/SC prevede che il circuito integrato della Smart Card (ICC) debba aderire all'ISO7816 nelle parti 1-2-3-10 . Allo stesso modo il lettore di Smart Card (IFD) deve aderire all'ISO7816 nelle stesse parti. E' anche utile sapere che una Smart Card al suo interno, ha una EEPROM con un livello logico strutturalmente analogo ad un File System di una memoria di massa generica; come lo è la sua unità di memoria lettura/scrittura. In essa sono memorizzate alcune elaborazioni eseguite dal processore, altre informazioni e programmi aggiuntivi (ad esempio le Applet) che arricchiscono le peculiarità predefinite di base della carta. Tanto per farla breve e non entrare in specifiche tecniche eccessivamente complesse diremo che PC/SC prevede tre metodi di accesso alle Smart Card

- Win 32: Tramite interfacce API di basso livello, che lasciano una grande libertà al programmatore ma necessitano della conoscenza approfondita di buona parte del sistema operativo presenta sulla Smart Card e degli altri dettagli tecnici
- CryptoAPI: Tramite interfacce API Crittografiche. Queste API implementano un alto livello per la crittografia delle informazioni, ma dipendono in modo molto forte da una particolare architettura e dalla tipologia di lettore/smart card. Attualmente in Windows le Cripto Api sono disponibili per alcuni modelli di Smart Card prodotti da Gemplus e da Schlumberger
- Scard COM: Si tratta di un'interfaccia non crittografica, di alto livello che "preleva" dalla Smart Card alcuni servizi e li rende disponibili al client

Per il nostro articolo utilizzeremo il primo metodo, che ci garantisce un'alta flessibilità ed è sufficientemente documentato in Windows.

Fig. 3: *Rappresentazione schematica dell'architettura PC/SC*

IL PLATFORM INVOKE

Le API Win32 rese disponibili da Windows sono contenute nella DLL winscard.dll. Chi segue ioProgrammo da un po' di tempo sa già che è possibile invocare una DLL unmanaged dall'interno del .NET framework con una tecnologia nota come Platform Invoke. Un esempio generico di Platform Invoke è il seguente:

```
Imports System
Imports System.Runtime.InteropServices
<DllImport("libreria.dll",
                    EntryPoint:="nomeFunzione", _
        SetLastError:=True, ExactSpelling:=True, _
CallingConvention:=CallingConvention.StdCall)> _
Public Function nomeFunzione(parametri, ecc ..
                                    parametro )
…
End Function
```

Per quanto riguarda in particolare la winscard.dll potremo utilizzare qualcosa del genere:

```
[DllImport("WinScard.dll")]
public static extern int SCardEstablishContext(int
     dwScope, int nNotUsed1,int nNotUsed2, ref int
                                    phContext);
```

Impareremo il significato delle varie funzioni esposte dalla DLL in questione man mano che andremo avanti nella comprensione della tecnica.

ACCESSO ALLA SMART CARD

Prima di tutto cerchiamo di capire quali sono i passi logici che andremo ad effettuare. Potremo riepilogare il flusso delle operazioni come segue:

1) Stabilire il contesto del "resource manager". Il resource manager si occupa di gestire l'accesso alle risorse collegate al computer. E' sostanzialmente una sorta di motore a cui fare riferimento per ottenere l'accesso ad eventuali lettori collegati. Stabilirne il contesto significa ottenere un riferimento da poter utilizzare per poterlo invocare. Ad esempio sarà il resource manager che ci indicherà se ci sono lettori connessi, quanti sono, e qual è il loro stato. Ogni volta che invochiamo un metodo della winscard.dll, dovremo passargli il riferimento del resource manager, di modo che il metodo in questione sappia a chi si deve rivolgere per ottenere le informazioni che gli servono. Un esempio in C# di come ottenere il resource manager è il seguente:

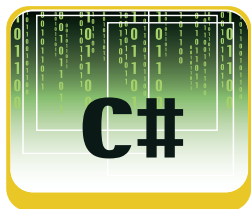
```
private void Form1_Load(object
                    sender, System.EventArgs e)
{
        uint nContext = 2;
        int nNotUsed1 = 0;
        int nNotUsed2 = 0;
        this.nContext = 0;
        string error = "";
        int nRetVal1 =
     SCardEstablishContext(nContext, nNotUsed1,
                    nNotUsed2, ref this.nContext);
        if (nRetVal1 != 0)
        {
          error = "error";
        }
```

Cerchiamo anche di capire quali sono i parametri passati alla **SCardEstablishContext().** In particolare l'nContext rappresenta lo "scope" dell'applicazione e può valere

SCARD_SCOPE_USER
SCARD_SCOPE_SYSTEM

Nel primo caso il resource manager potrà essere usato a livello di utente, nel secondo caso a livello di sistema. In questo secondo caso l'applicazione dovrà avere i permessi corretti. I parametri "nNotUsed1 e nNotUsed2, non sono attualmente implementati, servono per sviluppi futuri e nel nostro caso devono essere dunque settati a 0 o a null. Infine nContext conterrà l'handle generato come riferimento al resource manager e che come vedete abbiamo settato come globale all'applicazione, perché lo utilizzeremo come parametro per tutte le funzioni che utilizzeremo in futuro. Se la connessione al resource manager ha esito positivo otterremo un messaggio di SCARD_S_SUCCESS. Se fallisce otterremo un errore specifico. Un elenco è disponibile al seguente indirizzo: http://msdn.microsoft.com/library/en-us/secauthn/security/authentication_return_values.asp?FRAME=true#smart_card_return_values

2) Ottenere l'elenco dei lettori collegati al sistema. La funzione da utilizzare è la SCardListReader. E' possibile richiamarla con il platform invoke tramite il seguente codice:

```
public static extern int SCardListReaders
          (int hContext, string cGroups,ref string
             cReaderLists,  ref int nReaderCount);
int nRetVal2 =
SCardListReaderGroups(this.nContext,
                  ref cGroupList, ref nStringSize);
        if (nRetVal2 != 0)
        {
          error = "error";
        }
```

```
string[] cGroups = cGroupList.Split(delimiter);
string cReaderList = "" + Convert.ToChar(0);
int nReaderCount = -1;
int nRetVal4 = SCardListReaders(this.nContext,
    cGroups[0], ref cReaderList, ref nReaderCount);
if (nRetVal4 != 0)
{
    error = "error";
}
if (error == "") {
    string[] cReaders =
        cReaderList.Split(delimiter);
    toolStripStatusLabel1.Text =
        cReaders[0];
    this.readername = cReaders[0];
    connect();
    verifypin();
} else {
    if (MessageBox.Show("Errore di
  comunicazione lettore non presente " , "Error",
    MessageBoxButtons.OK,
  MessageBoxIcon.Question) == DialogResult.OK)
    {
        Application.Exit();
    }
}
```

Come potete anche vedere, abbiamo creato una funzione apposita per ricavare la stringa corretta dove saranno contenuti i nomi dei lettori. Anche in questo caso notate che abbiamo passato come parametro il contesto del resource manager.
Notate anche che nell'esempio in C# abbiamo associato this.readername = cReaders[0]; e subito dopo abbiamo richiamato connect() e verifypin().Queste due funzioni vengono richiamate se esiste almeno un lettore presente collegato al PC.

3) Connettersi alla smart card: in questo casola funzione da invocare è la SCardConnect. Via Platform Invoke si tratterà di utilizzarla come segue:

```
public static extern int SCardConnect
(int hContext, string cReaderName,
uint dwShareMode, uint dwPrefProtocol, ref int
        phCard, ref int ActiveProtocol);
```

Il nostro esempio in C# è il seguente:

```
private void connect(){
    uint nShareMode = 1;
    uint nPrefProtocol = 2;
    this.nCard = 0;
    this.nActiveProtocol = 0;
    int nRetVal6 =
  SCardConnect(this.nContext, this.readername,
  nShareMode, nPrefProtocol, ref this.nCard, ref
        this.nActiveProtocol);
    if (nRetVal6 != 0)
        DisplaySCError(nRetVal6,
            "SCardConnect()");
}
```

Notate che la SCardConnect prende come parametri il contesto del resource manager, il nome del lettore , il protocollo da usare per la trasmissione. In questo caso è stato scelto il trasferimento T0 che non specifica il protocollo utilizzato. Potevano essere specificati o il protocollo RAW o T1 mediante le costanti SCARD_PROTOCOL_T0 o SCARD_PROTOCOL_T1 ma questo lo lasciamo a voi perché dipende dal tipo di lettore usato.

LA FUNZIONE DISPLAYSCERROR

E' semplicemente una funzione di servizio che utilizziamo per mostrare un messaggio comprensibile, in seguito al verificarsi di una condizione d'errore. Un estratto della funzione in questione è la seguente:

```
private void DisplaySCError
    (int nErrCode, string cText)
{
    string cErrString = "";
    string ErrCode =
  String.Format("0x{0:X}", nErrCode);

    switch (ErrCode)
    {
        case "0x80100001":
            cErrString = "Internal
                Error";
            break;
        case "0x80100002":
            cErrString =
                "Cancelled";
            break;
[...]
if (MessageBox.Show("Errore di
  comunicazione "+ cErrString, "Error",
    MessageBoxButtons.OK,
    MessageBoxIcon.Question) ==
        DialogResult.OK)
    {
        Application.Exit();
    }
}
```

VERIFICHIAMO IL PIN

Per poter verificare il PIN , dobbiamo inviare alla card un comando ADPU. I Comandi ADPU rispondono allo standard ISO7816-4, tuttavia è sempre opportuno consultare la documentazione relativa al sistema operativo presente sulla Smart Card. Nel caso di infineon un breve estratto dei comandi più importanti è presente all'URL http://www.infineon.com/cgi-bin/ifx/portal/ep/channelView.do?channelId=-75271&channelPage=%2Fep%2Fchannel%2FproductCategories.jsp&pageTypeId=17224

Curiosando nella documentazione scopriamo che il comando per verificare la password di Amministratore della scheda è il seguente

```
00 20 00 02 08 31 32 33 34 35 36 37 38
```

E se non si verifica nessun errore il codice di risposta della scheda deve essere

```
90 00
```

Dove i numeri da 31 in poi rappresentano il PIN da passare alla scheda. Nel nostro caso 12345678. Per mandare un ADPU al sistema operativo è necessario utilizzare la funzione ScardTransmit. Il cui prototipo è definito come segue

```
[DllImport("WinScard.dll")]
    public static extern int SCardTransmit(int
    hCard, ref SCARD_IO_REQUEST pioSendPci,
                                        byte[]
        pbSendBuffer, int cbSendLength, ref
    SCARD_IO_REQUEST pioRecvPci, ref byte
        pbRecvBuffer, ref int pcbRecvLength);
```

Per brevità non trascriviamo l'intero codice per inviare l'ADPU, potete comunque trovare un riferimento nel codice allegato all'articolo. Quel che più conta è che sia chiaro quali sono i passi logici da compiere per arrivare a questo punto

SCRIVERE UN PROPRIO CODICE

Una volta che siamo autenticati con la password di amministratore, possiamo scrivere un nostro PIN utente nella scheda. Per farlo possiamo usare qualcosa del genere. L'esempio questa volta è in Visual Basic.NET

```
APDU(0) = &H0S
APDU(1) = &HD6S 'comando di scrittura UPDATE
                                   BINARY
APDU(2) = &H0S
APDU(3) = &H20S 'indirizzo della scrittura
APDU(4) = &HBS
'numero di bytes da scrivere B =
'11 in HEX che sarebbe "IoProgrammo"
'Imposto la lunghezza della stringa
'da scrivere nella  SmartCard
lenCode = 11
' adesso costruisco la stringa da inserire nella SC
InsStr2ArrByte(APDU, 5, "IoProgrammo")

'lunghezza dei dati della APDUa
lLenIn = 17
lLenOut = 2
lRet = SCardTransmit(hCard, ioInPci, APDU(0),
              lLenIn, ioInPci, ucBufOut(0), lLenOut)
```

Nel codice che abbiamo visto sopra, abbiamo creato una funzione che inserisce in un array di bytes, la stringa contenuta tra apici.
Questo vi faciliterà il compito quando dovrete scrivere dati prettamente alfanumerici.
Effettuata la scrittura del codice, come sempre controlleremo l'esito della risposta e successivamente, ci disconnetteremo dalla Smart Card con il metodo SCardDisconnect interrompendo l'alimentazione della carta:

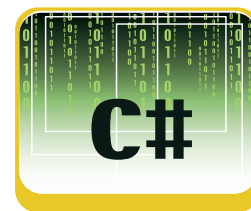
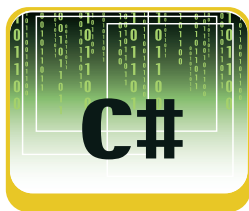
```
If (bBufOut(LenOut - 2) <> &H90S) And
              (bBufOut(LenOut - 1) <> &H0S) Then
   ListBox1.Items.Add("Scrittura dati nella Smart
                                  Card fallita")
   ListBox1.Items.Add("Numero errore: " & result)
   Exit Sub
End If
result = SCardDisconnect(hCard,
                          SCARD_UNPOWER_CARD)
```

LEGGIAMO IL CODICE SEGRETO

Adesso non ci resta che mostrare come leggere il codice che abbiamo scritto in preceden-

CLA	INS	P1	P2	LC	DATA	LE
0	*B0*	*0*	*20*	*05*	*0*	*0B*

za. Per far questo la procedura è molto simile alla scrittura. Se la connessione è andata a buon fine, dobbiamo occuparci della costruzione della APDU. La differenza sta nell'utilizzare il comando READ BINARY invece del UPDATE BINARY. Il READ BINARY è definito come segue: Il confronto dei dati avverrà poi nel vostro programma, comparando semplicemente le stringhe.
Analizziamo il codice:

```
'Il comando seguente è un comando di lettura.
'Legge  5 byte a partire dall'indirizzo &H20
bAPDU(0) = &H0S  'cla
bAPDU(1) = &HB0S 'ins (Instruction)
'(B0 determina  il comando di lettura)
bAPDU(2) = &H0S  'P1. byte alto
'dell'indirizzo di  lettura/scrittura
bAPDU(3) = &H20S
'P2. byte basso dell'indirizzo di lettura/scrittura
```

```
bAPDU(4) = &HBS
'LE (lunghezza in byte da  leggere) quindi B in
'esadecimale  = 11 per IoProgrammo
'parametro LC che definisce
'la lunghezza dei byte  inviati
LenIn = 5
'lunghezza dei byte ricevuti
'dalla funzione + 2 byte di risposta
'(SW1 e SW2)
LenOut = bAPDU(4) + 2
result = SCardTransmit(hCard, ioInPci, bAPDU(0),
                    LenIn, ioInPci, bBufOut(0), LenOut)
```

Una volta impostata l'APDU, ne constatiamo l'esito positivo. Anche qui come campi SW1 e SW2, dobbiamo aspettarci rispettivamente i valori 0x90 0x00. In caso non fossero tali, vorrebbe dire che il comando non è andato a buon fine.

```
'Controllo se i valori della risposta sono &h90 e
&H00 che contrassegnano la riuscita del comando
If (bBufOut(LenOut - 2) <> &H90S) And
                (bBufOut(LenOut - 1) <> &H0S) Then
   ListBox1.Items.Add
            ("Lettura dati non riuscita!")
Else
...
```

Se il tutto è OK, il codice è stato letto correttamente.

SDK E PACCHETTI DI TERZE PARTI

Abbiamo imparato i passi fondamentali da compiere per poter creare un semplice sistema di protezione, utilizzando le librerie fornite dal sistema operativo e senza acquistare alcun SDK. C'è da dire che ce ne sono molti in commercio e tutti ben fatti. Il problema principale è il loro costo. Quelli che vi possiamo consigliare, a basso costo ma con funzionalità davvero elevate, sono quello creato dalla Subsembly. Si integra benissimo con il sistema operativo e con l'ambiente di programmazione .NET e anche su sistemi operativi Windows CE. La miglior soluzione per rapporto prezzo-prestazioni. Uno che mi sento di consigliare, è senza alcun dubbio l'ASECard Memory Smart Card SDK venduto dalla ATHENA che comprende un lettore, le smart card e l'sdk per il loro sviluppo.
Il loro costo si basa sul tipo di SDK e sul tipo di smart card. Il kit è possibile acquistarlo anche presso la PartnerData s.r.l.
Oppure non spendere nulla e scegliere quello gratuito Microsoft.

Fig. 4: Il file system contenuto nella Smart Cards

CONCLUSIONI

Nella programmazione di applicazioni che fanno uso delle smart card, il problema principale resta legato ai diversi hardware e sistemi operativi supportati dalle schede. Nonostante si sia fatto un grande sforzo per ottenere degli standard affidabili, tuttavia nella pratica si riscontrano seri problemi nel far funzionare le applicazioni con ogni tipo di dispositivo e con ogni tipo di scheda.
In fase di produzione possiamo consigliarvi di testare il software sempre con lettori e schede specifiche e di rilasciarlo solo come conforme a quel particolare tipo di dispositivo. Il ciclo di programmazione risulta poi abbastanza semplice se si usa la winscard.dll. L-importante è avere capito quali sono i passi necessari per poter accedere in lettura/scrittura alla Smart Card.
La sintassi e la semantica delle funzioni esposte dalla winscard.dll è facilmente reperibile nella documentazione ufficuale Microsoft su http://msdn.microsoft.com
Abbiamo analizzato diverse problematiche relative alla protezione di un semplice software mediante una Smart Card.
Grazie al percorso intrapreso in questo articolo, sarete in grado di proteggere ogni tipo di software utilizzando il sistema descritto.
Se avete poi possibilità di spendere qualche soldo in più, abbiamo elencato due degli SDK più produttivi in circolazione.
Lasciamo a voi lettori il piacere di approfondire i temi trattati e di lasciar essere trasportati dalla curiosità, grande stimolo per noi programmatori.

DA XML A FLASH PASSANDO PER FLEX

UNA GUIDA PASSO PASSO ALLA NUOVA VERSIONE DEL PRODOTTO DI ADOBE DEDICATO AGLI SVILUPPATORI. VEDREMO COME, IN MODO RAPIDO E VISUALE, SI POSSANO SVILUPPARE INTERFACCE CHE FUNZIONANO SIA PER IL WEB SIA STANDALONE E PER TUTTE LE PIATTAFORME

Una volta c'erano i compilatori! Ovvero un tizio crea un file di testo con dentro delle istruzioni ben precise, lo dà in pasto ad un certo programma e quello ne tira fuori un eseguibile. È il principio base della programmazione. Tanto per capirci facciamo un esempio banale. Supponiamo di avere scritto con il notepad un fantastico file di testo contenente la seguente roba e di avere salvato questo file con il nome "pluto.mxml"

```
<mx:Button x="54" y="71" label="Button2"
                    click="clickHandler(event)"/>
```

Ora, supponiamo che sulla nostra macchina sia presente un compilatore che si chiama **mxmlc**. Potrei provare a dargli in pasto il mio file come segue:

```
mxmlc --strict=true --file-specs pluto.mxml
```

Ovviamente, quel che fa il compilatore è affare del compilatore! Potrebbe generare un eseguibile, un file HTML, un file in formato flash o java! Nello specifico Flex 2.0 è un compilatore. Prende in pasto un file XML creato con una certa sintassi e genera in output un file in formato Flash SWF. La cosa stuzzicante è che il file XML che diamo in pasto al compilatore può contenere oggetti quali bottoni, combobox, checkbox, funzioni, e handler di eventi. Capirete che la soluzione è molto interessante e per svariati motivi

1) È possibile generare applicazioni Flash che girano esattamente allo stesso modo sia sul Web che in modo Standalone
2) Le applicazioni Flash girano esattamente in modo identico sia su Linux che su Windows
3) Il formato flash consente di creare layout spettacolari

In buona sostanza è tutto quello che fa Java unito però alla semplicità e la potenza di un linguaggio che come vedremo ha una curva di apprendimento veramente bassa. I costi? Nulli! Il compilatore è assolutamente gratuito e liberamente scaricabile dal sito di Adobe. Se non volete spendere neanche i soldi della connessione ad Internet, lo trovate allegato nel CD di ioProgrammo. Se invece volete qualche comodità aggiuntiva, dovrete metter mano al portafoglio. Ma di questo parleremo meglio nel Box relativo a Flex Builder

IL PRIMO VAGITO

Iniziamo con creare un file chiamato con grande fantasia helloworld.mxml. Come già detto potete tranquillamente usare il notepad.
Riempiamolo con qualcosa del genere:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
     xmlns:mx="http://www.adobe.com/2006/mxml"
                         layout="absolute">
          <mx:Label x="177" y="172"
                    text="Hello World"
                    id="label1"
                    width="113"
                    fontSize="19"
                    fontFamily="Arial"
                    color="#ffffff"
                     fontWeight="bold"/>
</mx:Application>
```

Diamolo in pasto al compilatore come abbiamo già fatto all'inizio dell'articolo, nel modo seguente

```
mxmlc --strict=true --file-specs helloworld.mxml
```

Otterremo il file helloword.swf, cliccando sul quale vedremo nel player flash una finestra grigia al cui centro campeggerà la magica scritta "Hello World". Ovviamente possiamo anche provare a includere la nostra applicazione in un file html per renderla disponibile sul web, come

riportato nel seguente spezzone di codice

```
<noscript>
        <object classid="clsid:D27CDB6E-AE6D-
                        11cf-96B8-444553540000"
                id="pippo"
                width="100%" height="100%"
                codebase="http://fpdownload
                 .macromedia.com/get/flashplayer
                        /current/swflash.cab">
                <param name="movie"
                        value="pippo.swf" />
                <param name="quality"
                        value="high" />
                <param name="bgcolor"
                        value="#869ca7" />
                <param name="allowScriptAccess"
                        value="sameDomain" />
        <embed  src="pippo.swf" quality="high"
  bgcolor="#869ca7" width="100%"  height="100%"
          name="pippo" align="middle" play="true"
                  loop="false" quality="high"
                allowScriptAccess="sameDomain"
             type="application/x-shockwave-flash"
           pluginspage="http://www.adobe.com/go/
                             getflashplayer">
            </embed>
    </object>
</noscript>
```

Non è questo l'articolo che illustra come inserire una qualunque applicazione flash in un file HTML, tuttavia la tecnica da eseguire è esattamente quella classica.

CHE C'È IN QUEL FILE?

Le prime due righe sono composte come segue:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
                             layout="absolute">
```

Ovvero indichiamo che si tratta di un file in formato XML, instanziamo il primo "Container" ovvero l'applicazione. L'applicazione è il contenitore madre di tutti gli altri. Identificatelo se volete con la form. L'attributo **layout=absolute** ha più o meno lo stesso significato tipico dei layout di java. Ovvero indica l'allineamento che gli oggetti container devono seguire. Nel nostro caso abbiamo specificato che siamo noi a indicare la posizione in coordinate cartesiane dei vari oggetti che vogliamo disporre sulla form.
Il prossimo oggetto è più interessante. Si tratta di una label che abbiamo posizionato sulla form e identifichiamo con il nome label1

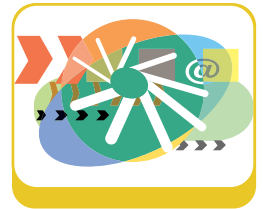
```
<mx:Label x="177" y="172"
   text="Hello World"
   id="label1"
```

omettiamo di parlare degli altri attributi per brevità e perché il loro significato ci pare piuttosto evidente.

GIOCHIAMO UN PO'

Quello che faremo adesso è aggiungere un bottone all'applicazione, premendo il quale la nostra label cambierà in "Siamo tutti amici di ioProgrammo". Il nostro codice varia come segue:

```
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
                             layout="absolute">
   <mx:Label text="Hello World"
           id="label1"
           width="113" fontSize="19"
           fontFamily="Arial"
           color="#ffffff"
           fontWeight="bold"
           x="177" y="127"
   />
   <mx:Button label="Button"
           x="201" y="194"
           id="button1"
           click="label1.text=&quot;
               Siamo tutti amici di ioProgrammo
          &quot;"
   />
</mx:Application>
```

Come vedete il codice è abbastanza semplice, non abbiamo fatto nient'altro che cambiare la property text della label utilizzando l'evento click sul bottone. Semplice, ma il codice è un po' sporco. Rendiamolo più elegante utilizzando una function e un gestore dell'evento clic. Quello che vedrete di seguito sicuramente susciterà un sorriso nei programmatori, perché lascerà intravedere tutte le infinite possibilità di

DI COSA ABBIAMO BISOGNO?

Prima di tutto del compilatore. Lo trovate all'indirizzo https://www.adobe.com/cfusion/tdrc/index.cfm?product=flex. Troverete sia la versione per Windows che la versione per Linux. In tutti e due i casi il prodotto è gratuito. Una volta terminata l'installazione avrete a disposizione fra le altre cose il magico "mxml" ovvero il compilatore che useremo per gli esempi di questo articolo

questo strumento, il codice è il seguente:

```
<mx:Button label="Button"
                x="201" y="194" id="button1"
                click="clickHandler(event)"
/>
<mx:Script>
        <![CDATA[
            import flash.events.MouseEvent;
            private function clickHandler (
                                    event:MouseEvent ):void
            {
                label1.text = "Siamo tutti amici di
                                            ioProgrammo";
            }
        ]]>
    </mx:Script>
</mx:Application>
```

Molto semplicemente intercettiamo l'evento Click sul bottone e in relazione a questo facciamo partire una funzione che prende come parametro l'evento stesso. La funzione non fa altro che modificare la property text della label. Quasi tutto quello di cui noi programmatori abbiamo bisogno è qui! Chi è abituato a programmare con un qualunque linguaggio non può non avere già familiarizzato con questi concetti. La curva di apprendimento di questo linguaggio è praticamente nulla!

COMPLICHIAMOCI UN PO' LA VITA

Quella che costruiremo adesso è un'applicazione banale, ma ci dà la possibilità di introdurre ancora qualche concetto su Flex. La nostra form adesso si comporrà di una stringa e di una combobox. Quando l'utente selezionerà un elemento della combobox la stringa varierà di conseguenza. Il codice mxml è il seguente:

```
<mx:Label text="Hello World"
        id="label1"
        width="407"
        fontSize="19" fontFamily="Arial"
        color="#ffffff" fontWeight="bold"
        textAlign="center"
        horizontalCenter="-6" verticalCenter="-38"
/>
<mx:Script source="script.as" />
<mx:ComboBox id="combolist"
                change="cambiaStringa()"
                horizontalCenter="0"
                verticalCenter="6">
    <mx:Array>
        <mx:Object data="0" label="Bello" />
        <mx:Object data="1" label="Brutto" />
        <mx:Object data="2" label="Completo"/>
        <mx:Object data="3" label="Grandioso"/>
        <mx:Object data="4" label="Meraviglioso" />
    </mx:Array>
</mx:ComboBox>
```

FLEX BUILDER

Adobe mette a disposizione un Plugin di Eclipse piuttosto potente che consente di sviluppare applicazioni Flex in modo rapido e visuale. Non solo il plugin è dota eclipse di Syntax Highlighting e Code Complexion per le applicazioni Flex, ma anche di un comodo Designer che fa si che le interfacce possano essere sviluppate in modo del tutto RAD. Flex Builder consente anche di compilare al volo le applicazioni ed è dotato di tutta una serie di Add-On che rendono davvero immediato lo sviluppo con Flex. Il prodotto non è gratuito ed è acquistabile al prezzo di 538

Vi invito a fare attenzione alla seguente riga:

```
<mx:Script source="script.as" />
```

Molto semplicemente in questo modo si indica che ingloberemo nel nostro file msxml le funzioni, le classi, le interfacce descritte con in linguaggio ActionScript all'interno del file script.as. Nel nostro esempio il codice all'interno del file in questione sarà

```
function cambiaStringa(){
 var stringa =  combolist.selectedLabel;
 label1.text="ioProgrammo è "+stringa;
}
```

La funzione **cambiaStringa()** verrà eseguita quando verrà intercettato l'evento change sulla combobox, grazie alla seguente linea

```
<mx:ComboBox id="combolist"
                change="cambiaStringa()"
                horizontalCenter="0"
                verticalCenter="6"
>
```

Ora, l'esempio è sicuramente banale. Ma i più attenti avranno intuito quali sono i vantaggi di un approccio del genere.

- Separazione della logica di business da quella di presentazione
- Disponibilità di un linguaggio ad oggetti di alto livello

Le possibilità a questo punto diventano infinite. Per completezza riportiamo un esempio con qualche costrutto più complesso. Questa

volta avremo una form che propone una domanda. Una listbox contenente tre risposte e infine un bottone. L'utente dovrà selezionare la risposta corretta dalla listbox e l'applicazione dovrà mostrare un messaggio che indica "hai vinto" se la risposta è corretta e "hai perso" se la risposta è errata. Vediamo il codice:

```
<mx:List id="miaList"  horizontalCenter="0"
                                verticalCenter="5">
       <mx:dataProvider>
          <mx:Object data="0" label="Paolino"/>
          <mx:Object data="1" label="Panino" />
          <mx:Object data="1" label="Poldino"/>
          <mx:Object data="1" label="Paperello"/>
          <mx:Object data="1" label="PincoPallo" />
       </mx:dataProvider>
</mx:List>
<mx:Button click="controllarisposta()"
                        label="Button"
                        horizontalCenter="0"
                        verticalCenter="138"
/>
<mx:Label
              text="Qual è il cognome di Paperino?"
              width="417" horizontalCenter="-2"
              verticalCenter="-113" textAlign="center"
              id="label1"
/>
```

E naturalmente il codice della funzione **controllarisposta**() contenuta in script.as

```
function controllarisposta():Boolean {
            var answer=miaList.selectedItem.data;
            if (answer==0) {
                         label1.text="Risposta Esatta";
            } else  {
                         label1.text="Risposta Errata";
            }
            return true;
}
```

Come vedete si tratta di un esempio decisamente semplice, tuttavia ci serve per capire quale sia l'interazione fra ActionScript e msxml.
Se avete fatto attenzione, avrete anche notato che il codice che abbiamo fin qui usato nello script.as è esattamente dello stesso tipo che nel primo esempio abbiamo usato inLine. ActionScript è il linguaggio di alto livello che ci consente di interagire con gli oggetti esposti da msxml.
L'esempio qui proposto ci dà anche la possibilità di introdurre un nuovo concetto: quello di dataprovider.

INTERAGIRE CON I DATI

Fin qui avrete notato che i dati sono stati inseriti negli script in modo statico. Sia nel caso della combobox che nel caso della listbox, abbiamo scritto i dati direttamente nel codice.
Ovviamente questa non è una grande idea! Tipicamente i dati sono contenuti in un database. Deve esistere in flex un modo per interagire con i database. Ed in effetti esiste!.
Per questo primo esempio i nostri dati saranno contenuti in un file XML. Vedremo in seguito a quali altri tipi di database possiamo accedere con Flex. Considerate per ora le seguenti linee di codice:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
     xmlns:mx="http://www.adobe.com/2006/mxml"
                                layout="absolute">
          <mx:Script source="script.as" />
          <mx:Model id="q" source="risposte.xml" />
          <mx:List id="miaList"
                         labelField="label"
                         horizontalCenter="0"
                         verticalCenter="5"
                         dataProvider="{q.q}"
           />
           <mx:Button click="controllarisposta()"
                         label="Button"
                         horizontalCenter="0"
                         verticalCenter="138"
           />
           <mx:Label
                text="Qual è il cognome di Paperino?"
                width="417" horizontalCenter="-2"
                verticalCenter="-113"
                textAlign="center"
                id="label1"/>
</mx:Application>
```

Con il relativo file XML

```
<?xml version="1.0" encoding="iso-8859-1"?>
<questions>
      <q label="Paolino" data="0"/>
      <q label="Panino" data="1"/>
      <q label="Poldino" data="1"/>
      <q label="Paperello" data="1"/>
      <q label="Pincopallo" data="1"/>
</questions>
```

Utilizzando questa tecnica abbiamo stabilito una sorta di relazione stretta tra l'oggetto List e il file risposte.xml. Tutta la logica di "Binding" è contenuta in queste linee

```
<mx:Model id="q" source="risposte.xml" />
<mx:List id="miaList" labelField="label"
```

```
horizontalCenter="0" verticalCenter="5"
dataProvider="{q.q}"/>
```

Non ci soffermeremo in questo articolo sulla logica che sottende al databinding e su qual è la sintassi di questi metodi. Quello che più importa in questo momento è sapere che esiste la possibilità di utilizzare un DataProvider per recuperare i dati. L'esempio che abbiamo proposto per XML è significativo. Nelle prossime puntate di ioProgrammo avremo modo di soffermarci ulteriormente sull'argomento.

FLEX E GLI ALTRI

Fra le possibilità più interessanti offerte da Flex c'è quella di poter interagire con altri linguaggi. Iniziamo con qualche esempio di interazione con PHP. Consideriamo il seguente script PHP

```
$mysql = mysql_connect(DATABASE_SERVER,
                       DATABASE_USERNAME,
                       DATABASE_PASSWORD);
mysql_select_db( DATABASE_NAME );
$Query = "SELECT * from redazione";
$Result = mysql_query( $Query );
$Return = "<redattori>";
while ( $User = mysql_fetch_object( $Result ) )
{
    $Return .= "<redattore>
      <nome>" .$User->Nome."</nome>
      <cognome>".$User->Cognome."</cognome>
     </redattore>";
}
$Return .= "</redattori>";
mysql_free_result( $Result );
print ($Return)
?>
```

Non fa altro che connettersi ad un database, recuperare un elenco di utenti e stampare il risultato formattandolo come un file XML nella forma:

```
<redattori>
  <redattore>
    <nome>Fabio</nome>
    <cognome>Farnesi</cognome>
  </redattore>
  <redattore>
    <nome>Gianfranco</nome>
    <cognome>Forlino</cognome>
  </redattore>
  <redattore>
    <nome>Domenico</nome>
    <cognome>Pingitore</cognome>
  </redattore>
  <redattore>
    <nome>Domenico</nome>
    <cognome>Pingitore</cognome>
</redattore>
</redattori>
```

Il nostro scopo è creare un'applicazione Flash che stampa questi dati in una griglia prelevandoli dal risultato fornito dal file PHP.

Il codice mxml è il seguente:

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
                               layout="absolute"
         creationComplete="userRequest.send()">
    <mx:HTTPService id="userRequest"
     url="http://172.16.0.241/work/flex/index.php"
              useProxy="false" method="POST">
    </mx:HTTPService>
    <mx:DataGrid x="38" y="192" width="401"
        dataProvider="{userRequest.lastResult.
                     redattori.redattore}">
        <mx:columns>
            <mx:DataGridColumn
                      headerText="Nome"
                      dataField="nome"/>
            <mx:DataGridColumn
                      headerText="Cognome"
                      dataField="cognome"/>
        </mx:columns>
    </mx:DataGrid>
</mx:Application>
```

Anche in questo caso il funzionamento è intuitivo. Nella riga

```
<mx:Application
    xmlns:mx="http://www.adobe.com/2006/mxml"
                               layout="absolute"
         creationComplete="userRequest.send()">
```

Facciamo in modo che appena la creazione dell'applicazione è completata venga richia-

E LA SICUREZZA?

Per consentire ad un'applicazione Flex/Flash di richiamare un file presente su un web server esterno alla macchina su cui risiede l'applicazione, è necessario inserire nella root del web server un file crossdomain.xml che deve contenere le policy di permesso per le applicazioni flash. Un esempio è il seguente:

```
<?xml version="1.0"?>
<!DOCTYPE cross-domain-policy
                              SYSTEM
"http://www.macromedia.com/xml/dt
     ds/cross-domain-policy.dtd>
<cross-domain-policy>
     <allow-access-from
                  domain="*" />
</cross-domain-policy>
```

mato il metodo **send()** dell'oggetto userRequest. L'oggetto in questione richiama il file PHP che abbiamo precedentemente creato, con la riga:

```
<mx:HTTPService id="userRequest"
        url="http://172.16.0.241/work/flex/index.php"
                useProxy="false" method="POST">
```

Ed eventualmente gli passa dei dati tramite un POST. A questo punto instanziamo una Datagrid

```
<mx:DataGrid x="38" y="192" width="401"
     dataProvider="{userRequest.lastResult.redattori.
                                        redattore}">
```

Che viene riempita, grazie al dataprovidere, con i dati prelevati dalla struttura XML definita dal file PHP scendendo nell'albero XML fino alla gerarchia redattore.redattore. A questo punto non dobbiamo fare altro che riempire le colonne con i Field nome e cognome.

EFFETTUIAMO UN POST

Quello che faremo in questo paragrafo sarà estendere l'esempio precedente, consentendo all'utente di inserire un nuovo dato nel DB. Il primo passo sarà modificare leggermente il file PHP. Il codice da aggiungere è il seguente

```
if( $_POST["nome"] AND $_POST["cognome"])
{
     //add the user
     $Query = "INSERT INTO redazione VALUES ('',
           '$_POST[nome]', '$_POST[cognome]')";
          mail("jaco@localhost","pippo",$Query);
     $Result = mysql_query( $Query );
}
```

Mentre il file msxml diventa

```
<mx:Application
     xmlns:mx="http://www.adobe.com/2006/mxml"
                              layout="absolute"
          creationComplete="userRequest.send()">
     <mx:HTTPService id="userRequest"
        url="http://172.16.0.241/work/flex/index.php"
                useProxy="false" method="POST">
      <mx:request xmlns="">
           <nome>{thename.text}</nome>
           <cognome>{surname.text}</cognome>
      </mx:request>
   </mx:HTTPService>
      <mx:DataGrid x="38" y="192" width="401"
     dataProvider="{userRequest.lastResult.redattori.
                                        redattore}">
          <mx:columns>
               <mx:DataGridColumn
               headerText="Nome" dataField="nome"/>
               <mx:DataGridColumn
                              headerText="Cognome"
                              dataField="cognome"/>
          </mx:columns>
          </mx:DataGrid>
     <mx:TextInput x="38" y="38" id="thename"/>
     <mx:TextInput x="38" y="91" id="surname"/>
     <mx:Button x="38" y="147" label="Button"
                     click="userRequest.send()"/>
          </mx:Application>
```

Vediamo di comprendere le linee più oscure. In particolare

```
<mx:request xmlns="">
        <nome>{thename.text}</nome>
        <cognome>{surname.text}</cognome>
     </mx:request>
```

Queste due linee effettuano un "Bind" tra il contenuto di tue caselle di testo e le due variabili che verranno passate in POST.
Le seguenti due linee

```
<mx:TextInput x="38" y="38" id="thename"/>
<mx:TextInput x="38" y="91" id="surname"/>
```

Rappresentano le caselline di testo da riempire e che sono bindate con il metodo precedente. Infine

```
<mx:Button x="38" y="147" label="Button"
                     click="userRequest.send()"/>
```

Non fa altro che richiamare nuovamente la **send** di **userRequest** passandogli questa volta i parametri nome e cognome in POST. Quando la richiesta arriva al file PHP. Tutto quello che è necessario fare è controllare se c'è un qualche valore presente in POST e se questo valore esiste effettuare l'update sul database.

CONCLUSIONI

Per questa volta ci fermiamo qui. Flex2 è uno strumento assolutamente rivoluzionario che modifica molto il modo di concepire e pensare un'applicazione. Non solo il modo di programmare è sufficiente vicino a quello tradizionale, ma vengono rispettati anche i canoni di multipiattaforma e separazione dell'interfaccia dalla logica di business. Per molti versi si potrebbe dire che Flex ha raggiunto in pochissimo tempo quello che Java rincorre da anni.

PROGRAMMA AJAX SECONDO... GOOGLE!

AL GRAN NUMERO DI FRAMEWORK PER SVILUPPARE APPLICAZIONI WEB IN TECNOLOGIA AJAX NON POTEVA MANCARE UN TOOLKIT SVILUPPATO DAL GIGANTE DI INTERNET. IL GOOGLE WEB TOOLKIT SI PRESENTA CON QUALCHE CARTA IN PIÙ. VEDIAMO QUALE

AJAX (Asynchronous Javascript And XML) è senz'altro la tecnica del momento; oramai esistono framework per la scrittura di applicazioni Web che usano AJAX in tutti i maggiori linguaggi di programmazione. Google è in prima linea nello sviluppo di applicazioni AJAX (basti pensare che sia Google Earth che Google Maps l'adottano) e ha reso disponibile un framework a suo modo rivoluzionario: esso non espone metodi che "incapsulano" le funzionalità volute; piuttosto fornisce un nuovo Java Runtime Enviroment (JRE) per Java ed un insieme completo di API che permettono di realizzare applicazioni Java stand-alone. Lo sviluppo di queste applicazioni (e il debug) è possibile con un qualsiasi IDE, per esempio Eclipse, e, solo successivamente, un compilatore (fornito insieme al framework) trasforma l'intera applicazione Java in una equivalente applicazione Html + Javascript!

GOOGLE WEB TOOLKIT

Il framework si chiama Google Web Toolkit (d'ora in poi GWT), la cui pagina di riferimento è http://code.google.com/webtoolkit/; esso è disponibile sia per piattaforme Windows sia Linux. Si compone di una serie di librerie (per l'emulazione di un ambiente Java) e di un compilatore. Il risultato del compilatore è un'applicazione Web che fa uso solo di tecnologie client standard (DHTML) e che, eventualmente, può interagire con un server per il reperimento dei dati. Tale reperimento non avviene attraverso l'usuale navigazione (e quindi il reload di pagine) ma usando delle chiamate RPC (Remote Procedure Call) e l'aggiornamento della pagina corrente mostrata sul browser; questo fa sì che l'utente abbia la sensazione di usare un'applicazione standard (all'interno del browser) più che una usuale applicazione Web dove, di solito, si susseguono pagine diverse.

DOWNLOAD E INSTALLAZIONE

Dalla pagina http://code.google.com/webtoolkit/download.html è possibile eseguire il download del framework. Di seguito saranno mostrati l'installazione e l'uso della versione per Windows (quella per Linux è del tutto simile) e la creazione di un nuovo progetto con Eclipse. Una volta scompattato l'archivio compresso, si ha un ambiente come quello illustrato in **Figura 1**.

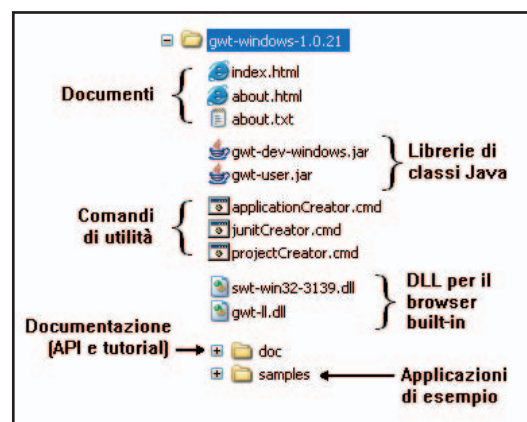



Fig. 1: la struttura del toolkit

GLI ESEMPI

Sotto la cartella samples/ è possibile accedere a degli esempi d'uso del framework. La loro struttura è la seguente:

a) nella cartella principale di ogni esempio ci sono due script: uno per eseguire la compilazione (ovvero per compilare le classi Java e trasformarle in file DHTML) ed uno per lanciare un browser particolare, chiamato "hosted";
b) una cartella contenente i file sorgente dell'esempio (la cartella è src/);
c) una cartella con i file .class risultato della compilazione dei sorgenti (cartella bin/);
d) una cartella contenente i file compilati in pagine DHTML (cartella www/);

UN PROGETTO IN ECLIPSE

Si supponga di voler utilizzare l'IDE Eclipse per la gestione di un progetto (si farà uso di Eclipse 3.1 con plug-in WebTool Project). In particolare si vuole far sì che il progetto contenga uno scheletro iniziale in cui sono impostati tutti i riferimenti alle librerie e in cui sia creata la classe principale, chiamata "modulo". Si apra una shell di comandi, ci si posizioni sulla cartella di installazione del GWT; quindi si digitino i comandi che seguono. Dapprima è necessario settare il path della cartella workspace (dove si trova Eclipse nel vostro sistema):

```
set PATH_WORKSPACE=D:\path\to\eclipse\workspace
```

Poi bisogna settare il nome del progetto da creare (nell'esempio esso è IoProgrammoGwt):

```
set NOME_PRJ=IoProgrammoGwt
```

Quindi va settato il nome del package dei file sorgente:

```
set PACKAGE=it.ioprogrammo
```

La creazione vera e propria avviene invocando due script shell; il primo serve a creare la struttura del progetto:

```
projectCreator -eclipse %NOME_PRJ%
  -out %PATH_WORKSPACE%\%NOME_PRJ%
```

Il secondo serve a creare l'ossatura di base contenente dei file sorgente (da estendere per creare l'applicazione vera e propria):

```
applicationCreator -eclipse %NOME_PRJ%
  -out %PATH_WORKSPACE%\%NOME_PRJ%
    %PACKAGE%.client.IoProgrammoGwt
```

Aprendo la cartella specificata precedentemente in PATH_WORKSPACE si può osservare che è stata creata un'applicazione con la struttura mostrata in Figura 2.

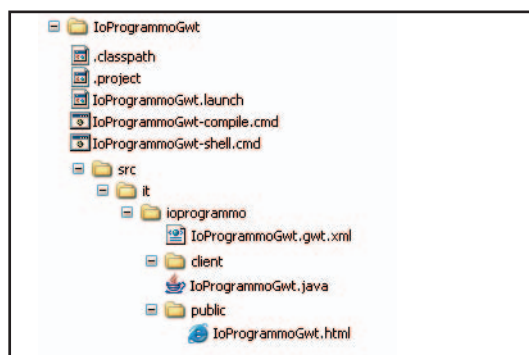



Fig. 2: *la struttura dell'applicazione creata*

Non resta che aprire Eclipse; quindi scegliere New > Project > Java Project. Nella finestra di dialogo specificare lo stesso nome inserito nella variabile NOME_PRJ. Eclipse si accorge che tale progetto esiste e avverte che verrà incluso. Basterà premere sul pulsante "Finish" per avere il nuovo progetto incluso tra quelli aperti.
Non resta che eseguire lo scheletro così creato (infatti, benché minimale, è stata creata un'applicazione completa). Per farlo si scelga Run > Run.... Nella finestra di dialogo ci dovrebbe essere IoProgrammoGwt, come mostrato in Figura 3.

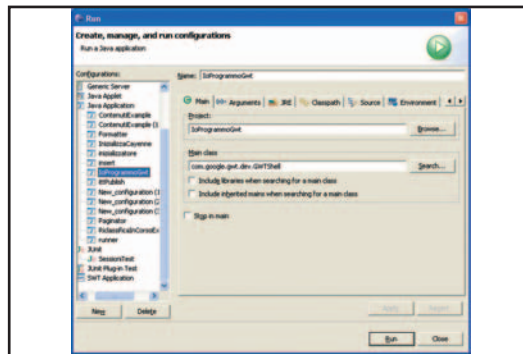



Fig. 3: *finestra di dialogo per eseguire l'applicazione*

L'esecuzione dell'applicazione consiste nell'aprire il browser hosted e l'applicazione appena creata (Figura 4).

Fig. 4: *esecuzione dell'applicazione creata*

LE VERSIONI DEL TOOLKIT

Al momento di scrivere l'articolo la versione disponibile del GWT è la 1.0.21, che è considerata ancora una "beta release". Pertanto, non ancora adatta ad un ambiente di produzione.

SUPPORTO DI JAVA

GWT permette di compilare programmi scritti in Java 1.4.2 (o successive versioni). Vengono emulate le classi contenute nei package java.lang e java.util (per la cui documentazione specifica, comprese eventuali limitazioni, si rimanda alla documentazione delle API). Esistono alcune limitazioni anche sui costrutti Java supportati ma che sono, in genere, del tutto marginali. Per esempio non è possibile utilizzare i metodi di reflection propri di Java (call by name, introspezione e altro); inoltre i tipi di dato a 64 bit sono "simulati" con tipi JavaScript a doppia precisione; si raccomanda anche una certa cautela nell'uso delle espressioni regolari, in quanto esse sono gestite in maniera leggermente diversa tra Java e JavaScript. GWT, infine, non supporta la serializzazione delle classi e le asserzioni e i costrutti di sincronizzazione sono ignorati nel codice risultante (perché il programma JavaScript ottenuto non è mai multithread, essendo eseguito su un unico client alla volta).

Verificato che tutto è andato a buon fine si può procedere nell'ispezione del codice sorgente creato in automatico e, successivamente, si può personalizzarlo. Per esempio si provi ad aprire il file it.ioprogrammo.client.IoProgrammoGwt.java e a cambiare il contenuto delle scritte (mettendole, per esempio, in italiano). Rieseguendo il run del programma ecco che appaiono subito le nuove modifiche.

IL PATH DI INSTALLAZIONE

I comandi contenuti in GWT per la creazione del progetto in Eclipse inseriscono, tra le proprietà del progetto, dei riferimenti alla cartella di installazione del toolkit stesso. Se si desidera procedere alla creazione ex-novo del progetto, seguendo i passi descritti nell'articolo, lo si può installare dove si preferisce; viceversa, per minimizzare i cambiamenti al progetto esistente, si consiglia di installare il toolkit nella cartella c:\gwt-windows\ (oppure intervenire da Eclipse usando Project > Properties > Java Build Path > Libraries e sui file .cmd e .launch). Inoltre, di default, il progetto è settato per essere eseguito in locale (senza comunicare via RPC con una parte server). Per cambiare questo comportamento si deve intervenire sulla classe IoProgrammoGwt.java e impostare a "true" la variabile SERVER_SIDE e inserire in SERVER_SERVICE_URL la url completa a cui risponde il servizio lato server.

NOTA

USIAMO IL SINGLETON

Il pattern Singleton permette di creare un unico oggetto di una classe e di riferirlo nelle altre classi senza creare nuovi oggetti (è pattern molto usato nelle applicazioni GWT).

ANALISI DEL CODICE SORGENTE

Il vantaggio dell'uso di un IDE come Eclipse si apprezza da subito: basta selezionare (doppio click) una qualsiasi delle classi o metodi (per esempio Button) e poi premere con il pulsante destro e scegliere, dal menu contestuale, "Open Declaration": subito appare la definizione della classe (o del metodo). In questo modo si può ispezionare il codice esistente e capirne il contenuto "by example". In particolare è possibile verificare che la classe IoProgrammoGwt implementa l'interfaccia EntryPoint, la quale ha un unico metodo: onModuleLoad(). Leggendo la documentazione si scopre che questo è il metodo invocato quando il modulo viene caricato (nel linguaggio GWT un modulo è un'applicazione).

Si può anche osservare che alla classe Button, come a tutte le altre classi che sono componenti dell'interfaccia, è possibile associare dei listener; in pratica il listener viene invocato al verificarsi di particolari eventi (se si usa un ClickListener, esso intercetta la pressione di un tasto del mouse, FocusListener quando riceve il focus e KeyboardListener intercetta eventi proveniente dai tasti della tastiera). Inoltre ciascun componente creato può essere agganciato ad uno o più componenti (pannelli o altricontenitori) ma tutti, alla fine, dovranno essere agganciati al componente RootPanel. Ma a chi si riferisce questo RootPanel? Esso non è altro che la pagina HTML di partenza. È interessante notare che l'esempio creato di default fa riferimento a due entità specifiche ("slot1" e "slot2") a cui aggancia i componenti creati:

```
RootPanel.get("slot1").add(button);
RootPanel.get("slot2").add(label);
```

Basta aprire il file it.ioprogrammo.public.IoProgrammoGwt.html e ispezionare il codice per accorgersi che essi fanno riferimento a due entità html contrassegnate con un ID omonimo:

```
<table align=center>
 <tr>
  <td id="slot1"></td><td id="slot2"></td>
 </tr>
</table>
```

Se, anziché riempire una tabella esistente (e fissa!) si vuole generare tutto il contenuto dinamicamente, il body html deve essere vuoto.
Un'altra cosa interessante, che deriva dall'analisi del file html, è che il link tra questa pagina (che rappresenta la pagina inziale e una sorta di "template" per la parte AJAX vera e propria) e il relativo modulo GWT è dato da un tag meta presente nell'intestazione:

```
<meta name='gwt:module'
 content='it.ioprogrammo.IoProgrammoGwt'>
```

LA NUOVA APPLICAZIONE

È giunto il momento di creare un'applicazione del tutto nuova. Questa prevede la creazione di un visualizzatore di... questo articolo! Ovvero si vuole un menu a sinistra con i titoli delle diverse sezioni: facendovi clic sulla destra appare il testo completo relativo alla sezione selezionata. In alto una semplice intestazione con il titolo e l'autore, insieme al catenaccio. Il nuovo metodo **onModuleLoad** potremmo riscriverlo come segue:

```
 public void onModuleLoad() {
   DockPanel main = new DockPanel();
   main.add(getTitolo() , DockPanel.NORTH);
   main.add(getAutore() , DockPanel.NORTH);
   main.add(getCatenaccio() ,
                         DockPanel.NORTH);
   main.add(getMenuSinistra() ,
                          DockPanel.WEST);
   main.add(getSezioniDestra(),
```

```
                DockPanel.CENTER);
   main.setWidth("100%");
   RootPanel.get().add(main);
}
```

In pratica si usa un pannello (DockPanel) a cui è possibile "attaccare" nuovi widget grafici su uno dei bordi (usando le opportune costanti che indicano i quattro punti cardinali) e l'ultimo widget agganciato prenderà la parte restante al centro. Il metodo getTitolo reperisce il titolo come immagine, getAutore, l'autore come testo HTML e getCartenaccio come Label (per i dettagli dei metodi si veda il codice sorgente allegato). Più interessanti i metodi **getMenuSinistra()** e getSezioni **Destra()** che, rispettivamente, mostrano la lista delle sezioni, ognuna con un'immagine di riferimento, e il testo completo della sezione selezionata.
Il menu a sinistra viene realizzato mediante un oggetto di tipo Tree (albero in cui i nodi sono label più immagine). L'oggetto che visualizza il testo può essere un semplice testo html con la possibilità di scroll. Inoltre è necessario far sì che un clic sul menu a sinistra coincida con il cambio del testo visualizzato nella sezione a destra. Per far questo è necessario eseguire un design particolare per l'applicazione (se si fa riferimento agli esempi standard è, più o meno, quanto realizzato nell'esempio Mail).

SINGLETON

Si possono creare due classi distinte, una che implementi la lista delle sezioni (quella a sinistra) e una che implementi la visualizzazione della sezione corrente (al centro). Chiamiamo la prima ListaSezioni. Essa estenderà la classe Composite, ovvero una classe che permette di eseguire il wrapping di un qualsiasi componente: in questo caso si vuole fare il wrapping di un oggetto Tree (che pertanto verrà dichiarato come attributo privato della classe stessa). Inoltre ListaSezioni implementerà l'interfaccia TreeListener, in quanto si vuole eseguire l'aggiornamento del testo mostrato al centro quando c'è un clic su uno degli elementi del Tree. Il costruttore della classe prenderà la lista delle sezioni e le disegnerà come link associati all'oggetto Tree (per comodità tutte le informazioni specifiche alle sezioni, ovvero titoli e testo associato, sono state racchiuse in una classe di utilità chiamata Sezioni).
Ora è necessario realizzare il metodo che intercetti il clic su un titolo e agisca sul componente al centro. Come far interagire i due? Un modo è il seguente: far sì che il modulo principale (IoProgrammoGwt) diventi un singleton, sia referenziato dal gestore dell'evento e invochi un apposito metodo di aggiornamento:

```
public void onTreeItemSelected(TreeItem item) {
  IoProgrammoGwt.getInstance().
   setSezione( testo );
}
```

A questo punto il metodo setSezione non farà altro che invocare un metodo della seconda classe, quella responsabile del disegno a video del testo (classe MostraSezione.java) a cui passa il testo inserito dal gestore dell'evento clic.
Non resta che eseguire il progetto e verificare che l'applicazione si comporti nel modo voluto.

PER ORA È SOLO CLIENT!

L'applicazione appena sviluppata in realtà non interagisce mai con un server remoto; infatti la classe Sezioni.java memorizza al suo interno tutte le informazioni riguardanti i titoli e i testi delle sezioni. Questa va bene se si vuole eseguire un test delle funzionalità client. Di solito invece, per applicazioni complesse, questa parte dovrebbe reperire le informazioni su un server attraverso chiamate RPC. GWT impone una struttura rigida per creare le classi lato client e lato server che devono essere messe in comunicazione via chiamate RPC. Questa "rigidità" è dovuta al fatto che lo sviluppatore si concentra sulle funzionalità, mentre è compito del toolkit (grazie alla reflection) creare il meccanismo di invocazione RPC, con relativa serializzazione /deserializzazione dei parametri e quant'altro è necessario per il suo corretto funzionamento. Ecco come procedere per comunicare i dati da client a server (e viceversa).

RPC: INTERFACCE E CLASSI SUL CLIENT

La prima cosa da realizzare è un'interfaccia. Essa deve estendere la classe RemoteService e dichiarare tutti i metodi che si vuole invocare sul server; per il nostro esempio essa può essere la seguente:

```
public interface SezioniService extends
                                 RemoteService{
  public String[] getTitoli();
  public String getSezione(String titolo);
}
```

Questa interfaccia rappresenta il "contratto" del servizio; come tale dovrà, successivamente, essere

NOTA

ECLIPSE ED I PATH

Per impostare dei path personalizzati su Eclipse, si usi Project > Properties > Java Build Path > Libraries; inoltre intervenire sui file .cmd e .launch

NOTA

ATTENTI A TOMCAT

La libreria gwt-user.jar ridefinisce la gerarchia javax.servlet. Per questo non può essere inclusa in una webapp ma deve essere resa accessibile nella cartella $TOMCAT_HOME/common/endorsed/

NOTA

E LA PRIVACY?

Monitorando le connessioni in uscita ci si accorge che la libreria comunica dei dati verso un server di Google; così verifica la presenza di aggiornamente e aggiorna le statistiche d'uso della libreria.

importata anche sulla parte server. Il client deve definire una seconda interfaccia; essa deve avere lo stesso nome della precedente ma con suffisso Async; inoltre i metodi sono gli stessi di quelli della classe precedente ma, questa volta, non restituiscono alcun valore e hanno un parametro in più, di tipo AsyncCallback:

```
public interface SezioniServiceAsync {
  public void getTitoli(AsyncCallback callback);
  public void getSezione(String titolo, AsyncCallback
                                          callback);
}
```

Per invocare un metodo sul server, è necessario costruire un opportuno oggetto dell'ultima interfaccia (SezioniServiceAsync) e, per costruirlo, si deve utilizzare una apposita factory (GWT.create) a cui si passa la classe dell'interfaccia SezioniService:

```
SezioniServiceAsync sez =
  (SezioniServiceAsync)
              GWT.create(SezioniService.class);
```

Inoltre si deve impostare l'end-point (che è una url espressa come stringa di caratteri) a cui risponde il servizio:

```
ServiceDefTarget endpoint = (ServiceDefTarget)
                                          sez;
endpoint.setServiceEntryPoint( qualeUrl );
```

Si vedrà in seguito come impostare questa url dell'end-point. Resta da effettuare l'invocazione RPC vera e propria. Tale invocazione è sempre asincrona, per quanto concerne la parte client. In particolare è necessario creare un oggetto AsyncCallback, nel quale ci sono due metodi: onSuccess (che viene invocato al completamento del metodo quando tutto è andato a buon fine) e onFailure (invocato, per esempio, quando il server non risponde o in tutte le situazioni "anomale"); ecco un esempio

```
AsyncCallback callback = new AsyncCallback() {
 public void onSuccess(Object result) {
IoProgrammoGwt.getInstance().setSezione
                                  (""+result);
 }
 public void onFailure(Throwable caught) {
  IoProgrammoGwt.getInstance().setSezione(
  "Impossibile contattare il server;"
    +caught.getMessage());
 }
};
```

Infine non resta che invocare il metodo asincrono passando gli opportuni parametri (l'ultimo dei quali è sempre l'oggetto AsyncCallback creato nelle righe precedenti, mentre i restanti parametri sono quelli utilizzati nell'interfaccia SezioniService):

```
sez.getSezione(valoreDaSettare, callback);
```

Per la parte client il lavoro è concluso. L'invocazione vera e propria viene interamente gestita dal framework. Non resta che realizzare le classi server che espongono il servizio richiesto; questa operazione è ancor più semplice di quanto fatto per la parte client...

TEST IN "LOCALE"

Avendo realizzato tutte le classi nel progetto originario, è possibile eseguire il test semplicemente inserendo un tag <servlet ... /> nel file di configurazione del progetto dove specificare, rispettivamente, la url della servlet che risponde in locale e la classe che la realizza; per l'esempio il file da personalizzare è src\it\ioprogrammo\IoProgrammoGwt.gwt.xml:

```
<module>
  <!-- ... altri tag -->
  <servlet path="/SezioniService"
   class="it.ioprogrammo.server.SezioniServiceImpl"
  />
</module>
```

Ovviamente nel path a cui far rispondere la servlet si può scegliere un qualsiasi nome. In ogni modo il valore di tale path è quello che va inserito come end-point del servizio sulla classe client che si occupa di creare l'oggetto SezioniServiceAsync. Ora è possibile rieseguire l'applicazione e verificare che funzioni anche in questa modalità.

TEST REMOTO

Benché il test in locale sia utile in fase di debug è necessario creare un progetto separato che realizzi la parte server del servizio e testarlo usando i due progetti come due applicazioni distinte (chiamiamo questa fase "test remoto"). Per farlo si crei un progetto "Dynamic

RPC: CLASSI SUL SERVER

La classe principale sul server (nell'esempio SezioniServiceImpl.java) è una servlet molto particolare; infatti essa deve estendere la classe RemoteServiceServlet e implementare l'interfaccia voluta definita in precedenza sul client (nell'esempio essa è SezioniService). Questa, SezioniServiceImpl.java, è l'unica classe da realizzare a livello di server (ovviamente si consiglia sempre di separare la logica di accesso ai dati su una classe ausiliaria, ma questo è un suggerimento di design che si può anche ignorare).

Web Project" con Eclipse e si copino le classi necessarie alla realizzazione della parte server create in precedenza (interfaccia SezioniService.java e la servlet SezioniServiceImpl.java); inoltre è necessario copiare il file gwt-user.jar nelle librerie del progetto. Attenzione al fatto che questo file non viene letto da Tomcat al momento del deploy! Infatti il jar contiene una ridefinizione della gerarchia javax.servlet e, per questo, ignorata dal container (per ragioni di sicurezza). Per questo motivo è indispensabile copiare il file jar nella cartella $TOMCAT_ HOME/common/endorsed/ prima di eseguire il deploy della webapp. L'ultima modifica necessaria è la personalizzazione del file web.xml affinché si dichiari la servlet del servizio e l'url a cui essa debba rispondere; per esempio:

```
<servlet>
  <servlet-name>SezioniService</servlet-name>
  <servlet-class>
   it.ioprogrammo.server.SezioniServiceImpl
  </servlet-class>
</servlet>
<servlet-mapping>
  <servlet-name>SezioniService</servlet-name>
  <url-pattern>/SezioniService</url-pattern>
</servlet-mapping>
```

Tutto a posto? Quasi! Bisogna modificare l'end-point sul client; in particolare è necessario inserire una url al servizio remoto (composta da nome del server o indirizzo ip, porta, nome applicazione e path a cui risponde la servlet); un esempio potrebbe essere http://127.0.0.1:8080/IoProgrammoGwtServer/SezioniService.

MONITORARE IL TEST

Siccome fidarsi è bene, ma non fidarsi è meglio, vogliamo curiosare soprattutto nella comunicazione tra l'applicazione e il server, si installerà un monitor; per esempio si può installare tcpTrace (http://www.pocketsoap.com/tcptrace/). Esso non fa altro che mettersi in ascolto su una porta e redirigere quanto riceve su un'altra (per esempio si può mettere Tomcat in ascolto sulla porta 8081, intercettare le richieste applicative indirizzate sulla porta 8080 e, con tcpTrace, redirigerle sulla porta 8081). In Figura 5 un esempio di comunicazione tra client e server; si possono notare alcune cose, come il fatto che la comunicazione è mantenuta attiva anche al termine dell'interazione e che i contenuti sono in formato binario per ottimizzare il flusso dei dati.

Fig. 5: ***tcpTrace ci rivela la comunicazione tra il client e il server.***

SI COMPILA!

Completato l'esempio non resta che compilarlo in codice DHTML. Per farlo basta mandare in esecuzione lo script chiamato IoProgrammo Gwt-compile.cmd (nella root del progetto). Esso crea automaticamente la cartella www/ con il codice compilato. Aprendolo con un qualsiasi browser appare l'applicazione precedentemente creata ma che fa uso, per la parte client, di solo codice DHTML (Figura 6).

Ivan Venuti

Fig. 6: ***l'applicazione, compilata in DHTML, come appare in Firefox***

NOTA

INTERFACCE CUSTOM

È possibile personalizzare il layout del codice DHTML generato intervenendo solo sui CSS. La documentazione delle API specifica gli stili associati ai diversi tag.

PER APPROFONDIRE...

Google offre un forum di discussione specifico per GWT (accessibile alla pagina http://groups.google.com/group/Google-Web-Toolkit) dove poter chiarire eventuali dubbi o esporre le difficoltà incontrate, e un blog (pagina http://googlewebtoolkit.blogspot.com/) dove è possibile restare aggiornati sulle ultime novità del framework. Essendo il framework ancora in versione beta, ci si aspetta un rapido susseguirsi di nuove versioni e, perché no, di funzionalità sempre più sofisticate. Stay tuned!

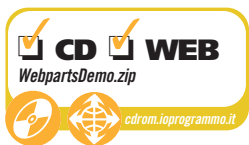
USARE IL BROWSER COME DESKTOP

UNA DELLE NOVITÀ PIÙ INTERESSANTI DEL FRAMEWORK .NET 2.0 È SICURAMENTE QUELLA DELLE "WEB PARTS". SI TRATTA DI UN METODO CHE CONSENTE AD UN UTENTE DI COSTRUIRE DINAMICAMENTE UNA PAGINA WEB ADATTANDOLA ALLE PROPRIE PREFERENZE....

Conoscenze richieste
Basi di Asp.net

Software
Visual Studio 2005, framework . NET 2.0

Impegno

Tempo di realizzazione

Sono due i nuovi elementi dell'interfaccia che recentemente stanno modificando radicalmente il concetto di applicazione web: Ajax e le Web Parts. La prima è una tecnologia che consente di creare siti che si aggiornano senza la necessità di Reload delle pagine. La seconda trasporta sul web il concetto di desktop tanto caro agli utilizzatori dei PC. L'idea è semplice, si deve pensare alla pagina web come una sorta di scrivania sulla quale disporre degli elementi. Gli elementi possono poi essere nascosti, spostati, manovrati come di consueto. Ad esempio immaginate una pagina web vuota, sulla quale l'utilizzatore (non il programmatore) può "poggiare" la lista dei suoi RSS preferiti. Se riuscite a visualizzare quest'attività avete già capito più o meno qual è il concetto di Web Parts.
Per meglio chiarire provate a fare una visita al sito live.com. In alto a sinistra è presente un menu che recita "imposta/nascondi", se lo cliccate nasconderete, oppure visualizzerete, la barra sottostante. Se, invece, provate a usare il pulsante: "Aggiungi Contenuti" scoprirete che è possibile aggiungere dinamicamente una "area" alla pagina. Ciascuna nuova "area" può racchiudere listbox, textbox, tabelle o qualsiasi elemento che compone una tipica pagina Web. In sostanza è possibile personalizzare l'Home Page di live.com aggiungendo, rimuovendo o spostando a piacimento queste particolari "aree", o per usare il termine tecnico: "Web Parts". Le Web Parts, presenti anche su Google ed Msn, sono un ulteriore passo avanti verso il Web 2!

WEB PARTS IN PRATICA

Asp.NET contiene una serie di controlli nativi che consentono di creare facilmente siti web che sfruttano profondamente le Web Parts.
Generalmente possiamo dire che il Team della nuova release di asp.NET si è ispirato ad uno stile di programmazione "dichiarativo" anche nel progettare per le Web Parts ponendosi come scopo, di ridurre al minimo il codice da scrivere e gli algoritmi da pensare e limitando il compito dello sviluppatore all'assemblaggio di componenti che, in maniera trasparente, fanno, semplicemente, quello che dichiarano di fare. All'interno di una tipica Web Form la sintassi per creare una Web Parts è semplice

```
<asp:WebPartZone ID="WebPartZone1"
                    Runat="server">
    <ZoneTemplate>
            Qui va inserito un qualsiasi
            elemento della pagina Web
    </ZoneTemplate>
</asp:WebPartZone>
```

Per essere precisi le Web Parts possono essere spostate solo in aree specifiche della finestra del browser denominate Web Zone; Nella Fig. 1, che illustra l'esempio allegato all'articolo, potete vedere diverse zone colorate che rappresentano appunto le Web Zone alle quali le Web Parts fanno riferimento.

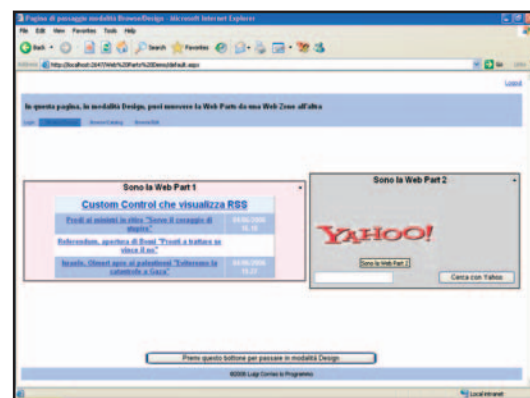



Fig. 1: ***Le zone colorate rappresentano le Web zone***

Le Web Part possono essere modificate, ovvero possono essere spostate, editate, minimizzate ed altro solo se lo stato della pagina lo

consente. Lo stato della pagina può essere programmato attraverso un evento innescato da un qualsiasi elemento dell'interfaccia utente: la pressione di un bottone, una voce selezionata da un DropDownList, eccetera.
Ad esempio mostriamo uno spezzone di codice che modifica lo stato della pagina in relazione al click su un bottone.

```
'Gestore di un evento legato ad un pulsante
    Protected Sub Button1_Click(ByVal sender As
      Object, ByVal e As System.EventArgs) Handles
                                     Button1.Click
        Try
            'La pagina e' in modalità Catalog
            WebPartManager1.DisplayMode =
              WebPartManager.CatalogDisplayMode
        Catch ex As Exception
            LabelError.Text =
                    Server.HtmlEncode(ex.Message)
            LabelError.Visible = True
        End Try
    End Sub
```

In ogni momento lo stato della pagina può essere in uno dei seguenti stati

Modalità Browse: Lo stato normale della pagina, nel quale nessun elemento può essere modificato
Modalità Design: In questo stato è possibile spostare le Web Parts attraverso le Web Zone
Modalità Catalog: In questo stato è possibile enumerare le Web Parts della pagina, aggiungerne ed eliminarne
Modalità Edit: In questo stato è possibile configurare le Web Parts, ovvero è possibile programmarne le dimensioni, le varie scritte, le relative icone, eccetera

INSERIRE O CANCELLARE

Immaginiamo l'Home Page di un moderno sito di News: ci sono Web Parts che illustrano le principali notizie di cronaca, politica, moda, scienza ed altro. L'utente può non essere interessato alla moda ma desiderare di essere informato sullo sport, è opportuno prevedere quindi la personalizzazione delle Web Parts presenti nella pagina. Quando la WebForm è in modalità Catalog appunto, è possibile ordinare le varie Web Parts presenti nella pagina utilizzando il controllo PageCatalogPart.
Questa funzionalità è ottenuta attraverso la maschera visibile in Fig. 2 che enumera le varie Web Parts presenti nella pagina, le aggiunge o le cancella;

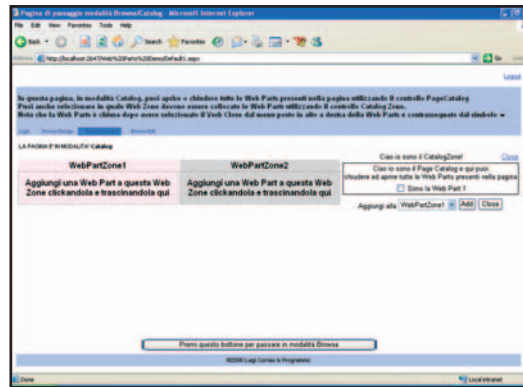



Fig. 2: ***Le web parts possono essere enumerate tramite un controllo di tipo PageCatalogPart***

```
<asp:CatalogZone ID="CatalogZone1"
                      Runat="server">
  <ZoneTemplate>
    <asp:PageCatalogPart
            Runat="server"
            ID="PageCatalogPart1" />
  </ZoneTemplate>
</asp:CatalogZone>
```

In alternativa è possibile utilizzare il controllo DeclarativeCatalogPart che consente allo sviluppatore di disegnare un Catalog personalizzato utilizzando le componenti preferite

```
<asp:DeclarativeCatalogPart
          ID="DCP1"
          runat="server">
    <WebPartsTemplate>
        Inserisci I tuoi controlli qui
    </WebPartsTemplate>
</asp:DeclarativeCatalogPart>
```

Nota che è possibile spostare Web Parts da una Web Zone all'altra non solo clickandovi sopra e trascinandole ma anche tramite il controllo PageCatalogPart perché questa funzionalità "Drag and Drop" non può essere implementata in tutti i modelli di browser.

WEB PARTS COME WINDOWS FORM

È possibile minimizzare chiudere, editare ed aprire la Web Parts attraverso un apposito menu, indicato nella Fig. 3 (nota la somiglianza con il tipico menu delle Windows dei sistemi operativi grafici); Concretamente questa funzionalità è implementata così:

```
<asp:WebPartZone ID="WebPartZone1"
                          Runat="server">
  <CloseVerb Text="Chiudi" />
  <HelpVerb Text="Help Online" />
  <ExportVerb Text="Esporta WebPart
                          Definition" />
  <DeleteVerb Text ="Cancella " />
  <MinimizeVerb Description="Minimizza" />
  <RestoreVerb Description="Ripristina" />
  <ZoneTemplate>
…
  </ZoneTemplate>
</asp:WebPartZone>
```

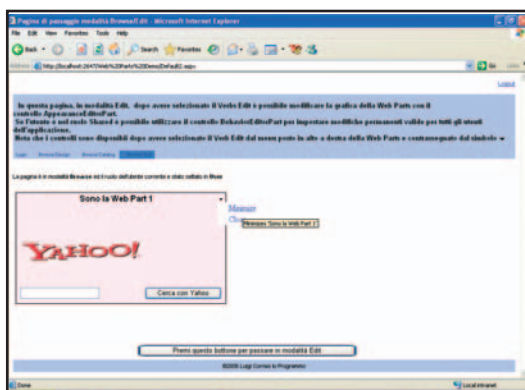



Fig. 3: ***Le Web Parts possono essere minimazzate o espanse tramite semplici pulsantini***

Il nome in codice degli eventi gestiti da questo menu è Verbs. Per la precisione, se la pagina non è nella statica modalità Browse, è possibile

- Minimizzare la Web Parts (MinimizeVerb).
- Riaprirla (RestoreVerb)
- Chiuderla (CloseVerb)
- Modificarla con i controlli dedicati che vedremo più avanti (EditVerb)
- Fare comunicare tra loro le Web Parts (ConnectVerb)
- Le Web Parts possono registrare la configurazione delle loro proprietà in un file XML che può essere importato in modo che i relativi settaggi siano ereditabili (ExportVerb)
- Lanciare una guida, ovvero una pagina web dedicata, che spieghi come funziona la Web Part (HelpVerb)

PERSONALIZZARE LA GRAFICA DELLE WEB PARTS

Quando lo stato della pagina è in modalità Edit, ed il Verb Edit è selezionato, esistono una serie di controlli appositi per la configurazione accurata della Web Parts

```
<asp:EditorZone ID="EditorZone1"
                    runat="server">
  <ZoneTemplate>
    <asp:AppearanceEditorPart
       ID="aep" Runat="server" />
    <asp:BehaviorEditorPart
               ID="bep"
               Runat="server" />
    <asp:LayoutEditorPart
               ID="lep"
               Runat="server" />
    Altri controlli…
  </ZoneTemplate>
</asp:EditorZone>
```

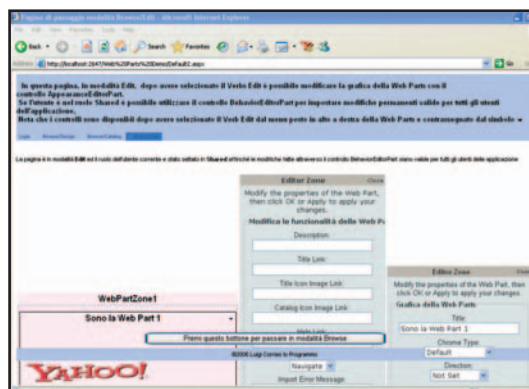



Fig. 4: ***Le Web Parts attraverso apposite maschere***

Questi marcatori generano le maschere illustrate in Fig. 4.
Per la precisione

- **AppearanceEditorPart** Serve a modificare l'altezza, la larghezza, la direzione (orizzontale o verticale) il titolo, il testo e il bordo
- **LayoutEditorPart** Serve a spostare le Web Parts, come se la pagina fosse in modalità Design, è utile perché alcuni browser non consentono di spostare le Web Parts trascinandole con il cursore del mouse
- **PropertyGridEditorPart** Utilizzato per pro-

Fig. 5: ***Il sito usabile.it è un buon riferimento***

grammare le proprietà delle Web Parts sviluppate da noi

- **BehaviorEditorPart** È un controllo che dovrebbe essere visibile solo agli utenti con il ruolo di amministratore dell'applicazione poiché le modifiche che imposta alla Web Parts saranno applicate a tutti gli utenti dell'applicazione; si occupa di selezionare quali Verbs è possibile abilitare.

Una curiosità: programmando le Web Parts, in modalità Edit, ci si imbatte nella non meglio definita proprietà Chrome;
questa proprietà si occupa della grafica del bordo che racchiude la Web Parts.
Nota che la parola Chrome viene utilizzata per definire la cornice che racchiude le finestra delle moderne interfacce utente grafiche.

Fig. 5: **Google fa un uso intensivo delle Web Parts**

IL RUOLO DEGLI UTENTI

Ogni applicazione Web moderatamente complessa ha delle pagine "amministrative" che consentono a utenti che godono di particolari permessi di gestire il sito web: ad esempio l'Amministratore di un Forum può cancellare alcuni account di utenti non graditi.
L'utente Amministratore ha il compito di personalizzare le Web Parts; Il ruolo di Amministratore è programmato nel file web.config

```
<webParts>
    <personalization>
        <authorization>
            <allow users="*"
                    verbs="enterSharedScope" />
        </authorization>
</personalization>
</webParts>
```

Una volta settato quali utenti hanno il ruolo di Amministratore è necessario impostare lo stato della pagina in maniera tale che sia

nello stato User, per l'utente Amministratore
nello stato Shared per tutti gli altri.

Questa funzionalità è programmata attraverso l'istruzione

```
' Innesca l'evento premendo un bottone
Protected Sub Toggle_Scope_Button_Click(ByVal
            sender As Object, ByVal e As EventArgs)
        _manager.Personalization.ToggleScope()
End Sub
```

che funge da interruttore tra lo stato della pagina User, ovvero lo stato dedicato all'Amministratore a quello Shared, lo stato di tutti gli altri utenti.
Creare un ruolo per gli utenti è indispensabile per utilizzare il controllo **BehaviorEditorPart** trattato poche righe sopra che, ripeto,si occupa di selezionare quali voci di menù saranno visibili a tutti gli utenti dell'applicazione.
Il ruolo degli utenti è memorizzato nel database Aspnetdb.mdf, un database SQL Server Express Edition, presente nella cartella App_Data e generato automaticamente dal .NET Framework allo scopo di conservare gli utenti delle nostre applicazioni ed i relativi ruoli.

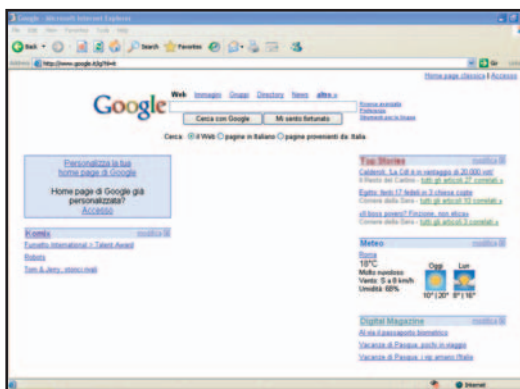



Fig. 6: **Notate oil riquadro celeste a sinistra**

WEB PARTS E USABILITÀ

Ogni volta che viene inventato un nuovo modo per assemblare una pagina web ecco che, a proposito o a sproposito, incominciano a proliferare siti che riprendono la nuova idea.
Se poi è veramente opportuno implementarla è un altro discorso;
decidere se le Web Parts sono adeguate al sito che stiamo costruendo non è solo una que-

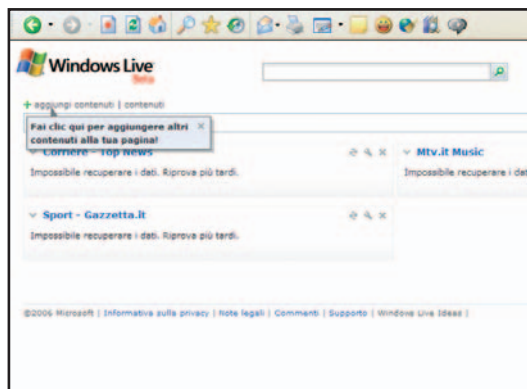



Fig. 7: **Il sito live.com è un buon esempio d'uso delle Web parts**

stione tecnica ma anche una questione di usabilità.
Per chiarire tutto questo abbiamo chiesto lumi a Maurizio Boscarol professore di Editoria Multimediale all'Università degli Studi di Trieste, autore del testo "Ecologia dei siti Web" e responsabile del sito usabile.it.
Per capire se le Web Parts sono usabili e quando è opportuno utilizzarle è necessario osservare l'uso che utenti reali fanno di un prodotto per capire le loro esigenze, magari attraverso test di usabilità per osservare direttamente le cose che loro stessi, anche se intervistati, non sarebbero magari in grado di esprimere.
Per l'utente l'usabilità delle Web Parts dipende essenzialmente da come l'utente stesso percepisce (o non percepisce) la possibilità di utilizzo.
A monte bisognerebbe capire se le personalizzazioni consentite sono viste dall'utente come un vantaggio o rimangono essenzialmente ignorate.
Non sempre la presenza di una possibilità tecnica si traduce in un vantaggio percepito.
È chiaro dunque che le possibilità di uso delle WP dipende essenzialmente dalla tipologia di utenti e dall'ambiente che si sta creando; ad esempio solo pochi fra gli utenti occasionali di un sito informativo si prenderanno la briga

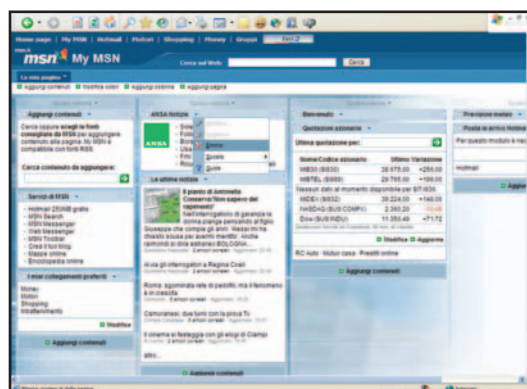



Fig. 8: **Un esempio d'uso sul sito msn.com**

di personalizzarlo: quello che cercano è la segnalazione delle novità più recenti, e un'interfaccia sufficientemente stabile da non dover essere reimparata ogni volta, al contrario, laddove il progetto è rivolto ad una serie magari chiusa di hard users, che usa ripetutamente il prodotto (penso al caso di una intranet, o di siti per utenti registrati, che consentano un'operatività ampia), allora le WP potrebbero trovare una buona applicazione
In ogni caso, proprio perché la risposta varia in base a contesti e a tipi di utenti e di compiti, il risultato andrebbe sempre preventivamente testato con utenti reali.
In sostanza, come ripeto, una possibilità tecnica è utile e proficua solo se inserita in un corretto processo di progettazione, che preveda verifiche con utenti.
Le possibilità funzionali senza una procedura di user centred design rischiano spesso di provocare più grattacapi agli utenti di quanti non ne contribuiscano a risolvere!

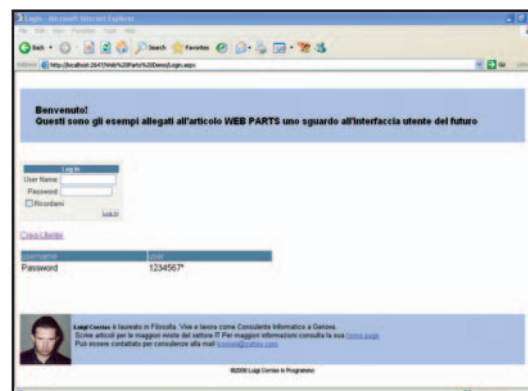



Fig. 9: **Il controllo sugli utenti è molto funzionale**

CONCLUSIONI

La brevità dell'articolo mi ha costretto ad accennare solamente alle molteplici funzionalità di questa tecnologia;
fedele al ritornello della canzone di Elvis Presley: "Little less conversation little more action" vi informo che, nell'esempio allegato, vedrete in azione tutto quanto esposto nell'articolo ed anche qualcosa in più…
Partendo da queste righe potrete senz'altro iniziare a costruire Web Parts professionalmente. La tecnologia è matura e Visual Studio contiene una serie di controlli che potranno ulteriormente aiutarvi. Il miglior modo per comprendere questo articolo è provare praticamente quanto descritto. La pratica mostrerà quanto questa tecnica sia efficace.

Luigi Corrias

DAL LINGUAGGIO SQL AGLI OGGETTI

INDAGHEREMO PIÙ A FONDO NELLE POTENZIALITÀ DEL FRAMEWORK HIBERNATE. PRESENTEREMO UN PLUGIN CHE CI PERMETTERÀ DI SVILUPPARE GUI ALL'INTERNO DI ECLIPSE. IMPAREREMO COME I DATABASE POSSANO ESSERE TRATTATI COME OGGETTI

Nel precedente articolo ci siamo occupati del framework Hibernate associato ad un RDBMS MySQL. In particolare abbiamo potuto apprezzare come, grazie a questo portentoso framework, sia possibile lavorare con un database proprio come se stessimo sviluppando una classica applicazione Object Oriented. Ottenere tutti questi vantaggi diventa ancor più agevole se come ambiente di sviluppo si utilizza Eclipse assieme al plugin messo a disposizione dalla stessa Hibernate Foundation (reperibile all'indirizzo http://www.hibernate.org/255.html .
Per chi si fosse perso la scorsa puntata facciamo un breve riassunto per focalizzare il punto da cui partiremo in questo articolo.

UN DATABASE AD OGGETTI

L'esempio da cui siamo partiti è l'implementazione di una semplice applicativo finalizzato alla gestione dei contenuti presenti nella nostra videoteca domestica.
Riportiamo quindi "l'analisi dei requisiti" effettuata la volta precedente tenendo a mente che essa costituisce la base su cui è stato progettato l'intero database e di conseguenza influirà pesantemente anche sul futuro sviluppo del nostro front-end:
La base di dati deve gestire l'elenco dei video in possesso dell'utente. Ogni video è memorizzato su uno o più supporti (nel caso di copie di backup) che debbono essere identificati da un numero univoco. Ogni video è identificato da un titolo e, per ciascuno di esso, possono essere specificate ulteriori informazioni come la data di uscita, il genere, il regista ed il cast di interpreti. Sia per i registi sia per gli interpreti è necessario specificare nome e cognome ed eventualmente la loro data di nascita. Per quanto riguarda il genere del filmato esso è identificato da un proprio nome (comico, drammatico, video clip, etc...).
Da quanto riportato sopra, in primo luogo, si è costruita la base di dati (il nostro back-end), il cui script SQL è riportato in allegato al CD.
Il passo immediatamente seguente è stato proprio quello di "riportare" il database in oggetti, o meglio in classi Java, in modo tale da avere, appunto, oggetti che mappassero la tabelle e le relazioni presenti nel back-end. Tale operazione non è risultata poi granché onerosa grazie all'utilizzo del plugin riportato all'inizio dell'articolo. In particolare gli strumenti di reverse engeenering messi a disposizione dal plugin di Hibernate ci ha permesso di avere un set di classi che ricalcano la struttura della base di dati su cui abbiamo impostato tutta l'applicazione.
Il diagramma UML di figura 1 riporta la struttura ottenuta (per motivi di brevità si riporta solamente tale rappresentazione grafica senza dilungarsi in ulteriori considerazioni fatte nell'articolo precedente). Il plugin in questione ci ha inoltre permesso di generare gli opportuni file di configurazione che permettono al framework Hibernate di comunicare con il database sottostante.
Ci eravamo infine lasciati con un paio di esempi che illustravano come, arrivati a questo punto, fosse molto semplice persistere i nostri oggetti sul relativo database. Sarà proprio da qui che ricominceremo.
Per essere più precisi divideremo questo articolo in due macro-sezioni:

1 prenderemo una maggiore dimestichezza con il framework di Hibernate implementando una serie di HQL Procedures (capiremo poi meglio di cosa si tratti) da invocare all'occorrenza da un'interfaccia grafica.

2 implementeremo, per mezzo di un altro plugin di Eclipse, una serie di form che faranno la nostra applicazione veramente user-friendly

NON PROPRIO SQL MA HQL

Come abbiamo ripetuto già più volte, Hibernate è sostanzialmente un framework logico che si frappone tra lo strato applicativo e quello del DB (vedi figura 2). È naturale quindi che, dal lato applicativo, non andremo ad utilizzare direttamente dei comandi SQL, sia perché ciò renderebbe inutile o superfluo l'utilizzo di tale framework, sia perché dobbiamo sempre tenere a mente che stiamo sviluppando in ottica Object Oriented e quindi non abbiamo più a che fare con tabelle e relazioni (fortunatamente!) ma con classi. D'altro canto è altrettanto logico che per “maneggiare” questo tipo di oggetti si utilizzerà qualcosa di molto vicino allo standard SQL.
Ciò di cui stiamo parlando è proprio HQL che sta proprio per Hibernate Query Language.
Senza dilungarci in tediose discussioni su quali siano le differenze sostanziali fra questi due paradigmi mettiamo subito in campo un paio di esempi utili per chiarirci le idee al riguardo.
Per prima cosa riproponiamo la classe HibernateUtil alla quale ci appoggeremo spesso:

```
public class HibernateUtil {
        private static final SessionFactory
                                        sessionFactory;
        static {

        sessionFactory = new Configuration()
                .configure().buildSessionFactory();
        }

        public static SessionFactory getSessionFactory()
                                                        {
                return sessionFactory;
        }
}
```

In sostanza questa classe ci permette di lavorare con un'unica session factory indipendentemente dal punto in cui viene invocato; in altre parole non abbiamo fatto altro che implementare un Singleton.
Quello che ora ci piacerebbe realizzare è una serie di procedure sulla falsa riga di quelle che sono le Stored Procedures implementabili ormai su quasi tutti i moderni RDBMS.
Partiamo dalla classe più semplice: CD.
Come potete notare, dal diagramma in figura 1, la classe CD ha un solo attributo di tipo int che identifica in maniera univoca il supporto indicato. Se volessimo creare una procedura che si occupi della persistenza di tale classe basterà:

```
public class CDProcedures {
.......
        public void insert(int n, Session session){
                Cd cd = new Cd();
                cd.setNumero(n);
                session.save(cd);
        }

.....
}
```

Quello che va sottolineato in questo caso è che l'oggetto Session ci è passato dal chiamante e quindi ci dobbiamo solamente occupare di scrivere le nostre azioni (nello specifico un salvataggio del cd per mezzo del metodo save) sull'oggetto Session. Quindi è bene tener presente che l'oggetto in questione non è stato ancora salvato sul database.
Se invece vogliamo creare una procedura one-shot, cioè una procedura che renda subito effettive le nostre azioni sul DB, dovremmo fornire un overload di tale metodo:

```
        public void insert(int n) {
                Cd cd = new Cd();
                cd.setNumero(n);
                Session session = HibernateUtil.ge
               SessionFactory().getCurrentSession();
                session.beginTransaction();
                session.save(cd);
                session.getTransaction().commit();
        }
```

In questo caso tutta la transazione viene gestita all'interno del metodo stesso, dal punto del begin fino al punto di commit.
Fin qui però abbiamo visto ben poco di qualcosa che somigliasse al buon vecchio SQL. Vediamo quindi ora una procedura che dal database “tiri su” tutti i CD a nostra disposizione e ce li esponga come un array della classe CD.
Il codice in questo caso è un po' più complesso del precedente.

```
public static Cd[] getAll() throws HibernateException
                                                        {
                Session s =
                 HibernateUtil.getSessionFactory()
                                  .getCurrentSession();
                try {
                        s.beginTransaction();
                        List res =
                     s.createQuery("from wrap.Cd")
                                                .list();
                        Cd[] out = new
                                        Cd[res.size()];
                    out=(Cd[])res.toArray(out);
                        s.close();
                        return out;
                } catch (HibernateException e)
                {
```

COSA VUOL DIRE WYSIWYG

L'acronimo WYSIWYG sta per What You See Is What You Get (quello che vedi è quello che hai) ed indica tutti quegli editor/IDE in cui si può ottenere una rappresentazione grafica immediata di ciò che si sta sviluppando.

NOTA

ALTRI PLUGIN DI ECLIPSE

La Eclipse Foundation ha sviluppato un plugin per la generazione di GUI di nome Visual Editor Project (http://www.eclipse.org/vep/WebContent/main.php) che però, a mio parere, risulta ancora un po' più pesante di Jigloo

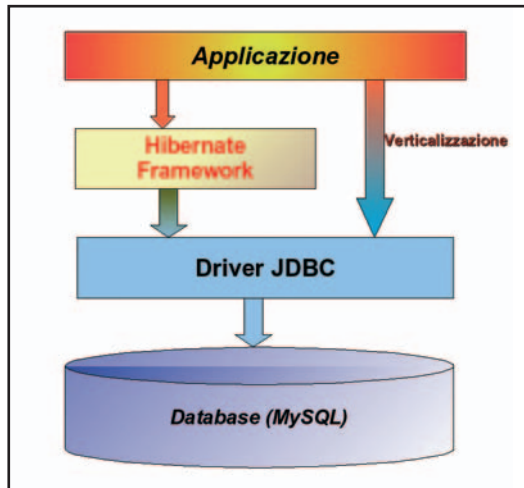



Fig. 2: **Architettura complessiva di un'applicazione basata su Hibernate**

```
                throw e;
        }
}
```

Cominciamo con il concentrarci sulla query. Il comando inviato al framework from wrap.Cd. Questa sintassi somiglia molto alla classica query SQL: SELECT * FROM Cd;
Visto che però abbiamo a che fare con classi e non con campi la prima parte di quella query, per adesso, risulterebbe superflua. Va inoltre notato anche come sia sempre necessario specificare il nome esteso della classe, cioè compreso il nome del package (nel nostro caso wrap).
Una volta creata la query si trasforma il risultato dell'interrogazione mediante il metodo list() che restituisce appunto un oggetto di tipo List. Poiché tale classe contiene generici Object dobbiamo effettuare un cast dinamico e contemporaneamente estrarre un array Cd. La soluzione qui adottata risulta la più efficiente sia a livello di performance sia a livello di pulizia di codice; eventuali cicli for o simili appesantirebbero solamente il tutto. Ora che abbiamo acquisito un po' più di confidenza con il fra-

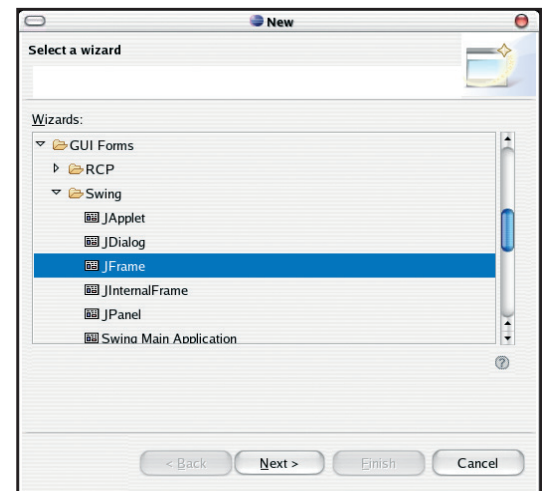



Fig. 3: **Schermata di creazione della Form.**

mework Hibernate introduciamo un altro plugin per Eclipse che ci permette di realizzare delle interfacce grafiche in maniera relativamente semplice.

COSTRUIRE GUI CON ECLIPSE? ORA È POSSIBILE CON JIGLOO

Uno dei punti deboli dell'IDE Eclipse, per lo meno nei confronti del suo diretto concorrente NetBeans della SUN, è stato proprio quello di un supporto WYSIWYG per lo sviluppo di interfacce grafiche. Fortunatamente l'architettura di Eclipse è nata per essere "scalata", cioè per essere ampliata mediante l'uso di opportuni plugin a seconda delle esigenze dello sviluppatore. Il plugin che andremo ad utilizzare si chiama Jigloo ed è disponibile all'indirizzo http://www.cloudgarden.com/jigloo/.
Dopo averlo installato, vediamo subito di metterlo all'opera per creare il Jframe principale per la nostra applicazione. Prima di tutto, dopo aver creato un nuovo package dove mettere tutti i componenti GUI, scegliere File ->New -> Other. A questo punto, se l'installazione del plugin è andata a buon fine, selezionare GUI Forms e la sotto-cartella SWING (visto che saranno questo tipo di librerie che andremo ad utilizzare). All'interno di tale sotto cartella selezionare infine JFrame ed immettere il nome per tal classe. Jigloo vi aprirà così una nuova prospctive simile a quella mostrata in figura 4. Naturalmente sarebbe troppo lungo ed anche noioso dilungarsi su ogni aspetto di questo plugin od

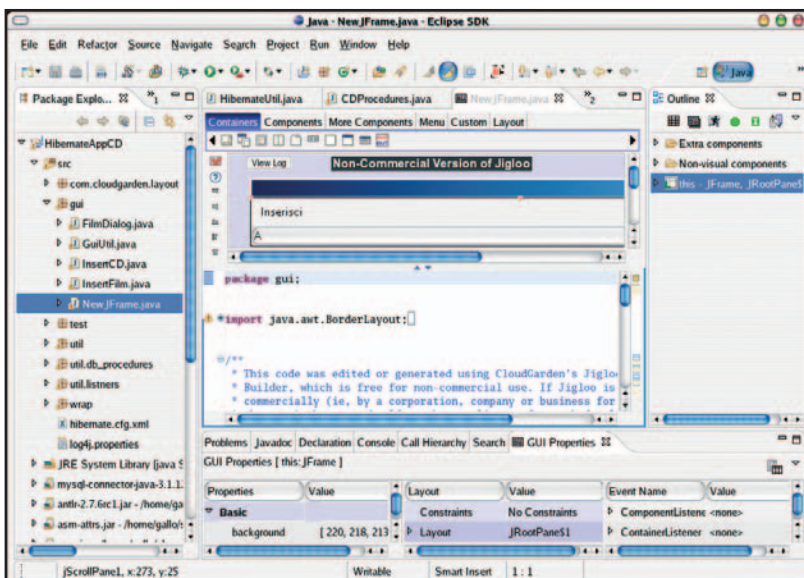



Fig. 4: **Prospective generata da Jigloo**

ogni componente dell'interfaccia che stiamo realizzando. Inoltre Jigloo si rivela un plugin abbastanza intuitivo per chiunque conosca un po' di Swing; per questo motivo elencheremo semplicemente tutti i componenti grafici che andremo ad assemblare insieme. Come avrete notato, Jigloo divide la schermata centrale a metà: la parte superiore vi mette a disposizione l'ambiente drug & drop per "sistemare" i vari componenti, mente quella inferiore riporta il codice generato da ogni vostra azione effettuata nella metà superiore. Il nostro JFrame deve essere composto da due JPannel affiancati orizzontalmente: il primo conterrà un menu che ci permetterà di aprire nuove form ed una JTable dove visualizzeremo i risultati delle nostre ricerche. Tali ricerche sono rese possibili grazie al secondo JPannel in cui sono presenti, oltre al bottone Cerca, il JField per il Titolo del video e due JComboBox per il numero di CD ed il Genere. Naturalmente a questi campi abbiamo affiancato le loro rispettive JLabel. Una volta composta la struttura sopra descritta e riportata in figura 5, bisogna mettere mano al codice per implementare tutte quella funzionalità che sarebbe impossibile dal lato WYSIWYG. La prima cosa da fare è definire il listner per il bottone Cerca, tale listner sarà la classe NewFrame stessa:

```
public class NewJFrame extends javax.swing.JFrame
        implements MouseListener,ActionListener {
......
jButtonCercaFilm.addMouseListener(this);
......
}
```

Occupiamoci poi della JTable. Questa classe ha bisogno di incapsulare un oggetto TableModel che gestisca i contenuti della Table. In altre parole bisogna implementare la classe astratta sopra citata nel seguente modo:

```
public class FrontTable extends AbstractTableModel {

        private Film[] films;

        public FrontTable(Film[] films){
                this.films=new Film[films.length];

      System.arraycopy(films,0,this.films,0,films.length);
}

        public int getRowCount() {
                return this.films.length;
        }

        public int getColumnCount() {
                return 1;
```

```
        }

        public Object getValueAt(int rowIndex, int
                                    columnIndex) {
                return films[rowIndex];

        }

}
```

I metodi che sono stati riportati sono quelli che sono definiti come astratti nella classe base e i loro nomi sono già autoesplicativi. La considerazione importante che in questo caso va fatta è quella di prestare attenzione a come sono incapsulati i dati. Infatti, come noterete, un array di Film viene copiato al momento dell'invocazione del costruttore ed in base ad esso vengo effettuate le operazioni sulla tabella stessa. Al momento dell'inizializzazione creiamo un FrontTable con un oggetto Film vuoto in modo tale da far apparire la JTable vuota al momento dell'avvio.

```
TableModel jTableOutModel = new FrontTable(new
                Film[]{
                        new Film()
                }
);
        jTableOut = new JTable();
        jScrollPane1.setViewportView(jTableOut);
        jTableOut.setModel(jTableOutModel);
```

Per quel che riguarda invece le ComboBox dobbiamo caricare da DB rispettivamente tutti i generi ed i CD presenti. Ad esempio per i CD tal compito viene svolto dal seguente metodo: (anche per i generi il

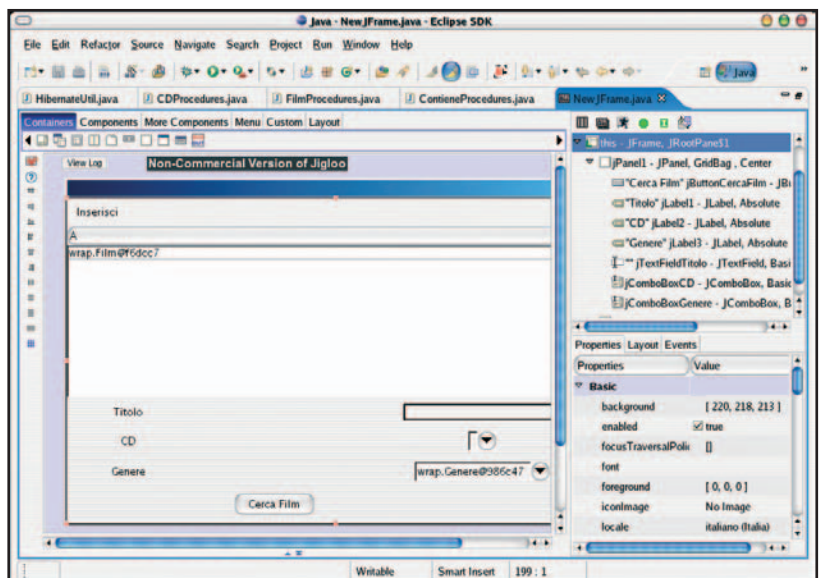



Fig. 5: ***Layout della classe NewJFrame.java dove sono anche elencati tutti i componenti.***

metodo è pressoché analogo)

```
        private Object[] getCD(){
                try {
                Cd[] cd =CDProcedures.getAll();
                Object[] obj = new
                                Object[cd.length+1];
                        obj[0]=new Empty();
System.arraycopy(cd,0,obj,1,cd.length);
                        return obj;
                } catch (HibernateException e) {
                        return newString[]{""};
                }
        }
```

Oltre ad invocare una procedura illustrata precedentemente, abbiamo aggiunto un oggetto empty all'inizio dell'array così che l'utente possa effettuare la ricerca indipendentemente dal CD. In altre parole se il numero di CD non è specificato la ricerca verrà effettuata su tutti i supporti di memorizzazione. Lo stesso vale per la classe Genere. Il caricamento sulla JComboBox avviene in questa maniera:

```
ComboBoxModel jComboBoxCDModel = new Default
                        ComboBoxModel(getCD());
jComboBoxCD = new JComboBox();
...
ComboBoxCD.setModel(jComboBoxCDModel);
```

A questo punto non ci rimane che scrivere il codice necessario ad effettuare una ricerca (che passi naturalmente per il framework Hibernate) dei contenuti multimediali che desideriamo. Tale codice verrà attivato nel momento in cui l'utente premerà il bottone Cerca, per cui si ha:

```
public void mouseClicked(MouseEvent e) {
                if(e.getSource()==jButtonCerc
                                        Film){
                        Film[] films = FilmProc
        dures.getFilms(jTextFieldTitolo.getText(),true);
                        jTableOut.setModel(
                        new FrontTable(films)
                        );
                }
```

INSTALLAZIONE DI JIGLOO

Per installare Jigloo basta selezionare dal menu di Eclipse Help->Software Updates->Find and Install e selezionare search for new features to install.Cliccare poi su New Remote Site ed immettere l'URL http://cloudgarden1.com/update-site, a questo punto sarà Eclipse stessa a scegliere la versione giusta per il vostro IDE. Per chi lavora su macchine POSIX ricordarsi che queste operazioni vanno eseguite dallo stesso utente che ha installato Eclipse (tipicamente il SU).

```
        }
```

Tutta la logica è quindi contenuta nel metodo statico FilmProcedures.getFilms che andiamo immediatamente ad analizzare:

```
        public static Film[] getFilms(String titolo,
                                        boolean lasco)
                {
                String where;
                if(lasco)
                        where ="f.titolo like
                                '%"+titolo+"%'";
                else
                        where ="f.titolo ="+titolo;
                String query = "from wrap.Film f
                                where "+where;
                Session s = HibernateUtil.
        getSessionFactory().getCurrentSession();
                s.beginTransaction();
                List list = s.createQuery
                                (query).list();
                s.close();
                Film[] films = new Film[list.size()];
                films=(Film[])list.toArray(films);
                return films;
        }
```

È stato definito un flag che indica se la ricerca va effettuata sulla frase esatta o basta che il titolo contenga la frase come sotto-stringa. Infatti la clausola where viene costruita in questo modo:

```
        if(lasco)
        where ="f.titolo like '%"+titolo+"%'";
                else
                        where ="f.titolo
                                ="+titolo;
```

dopodichè si crea la query vera e propria:

```
        String query = "from wrap.Film f where
                                "+where;
```

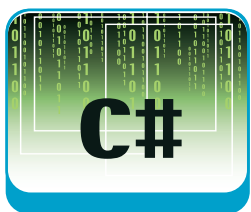
La parte restante del codice non presenta nulla di nuovo rispetto alle procedure già presentate. Lascio a voi, se volete, creare nuove procedure che tengano in considerazione ulteriori classi del package wrap.

CONCLUSIONI

Nell'arco di questi due articoli abbiamo avuto modo di apprezzare una parte delle potenzialità che ci vengono messe a disposizione da Hibernate, Eclipse ed i suoi plugin, ed il mondo Java in generale. Tutto naturalmente in piena filosofia Open Source si intende!

Andrea Galeazzi

CUSTOM DATA PROVIDER CON .NET

IMPLEMENTIAMO UN COMPONENTE INTERMEDIO CHE CONSENTE DI ACCEDERE A QUALUNQUE TIPO DI DATO UTILIZZANDO GLI STESSI METODI E INTERFACCE E VARIANDO SOLO POCHI PARAMETRI. SFRUTTEREMO IL NOSTRO HD COME UN DATABASE

Un .NET data provider è un insieme di classi utilizzato solitamente per connettersi a un database, eseguire comandi SQL su di esso e leggere i risultati di tali comandi. Esistono dunque provider specifici per i database più supportati, come Access ed SQL Server, e provider forniti eventualmente da terze parti, come ad esempio quelli che permettono l'accesso a MySql o PostgreSql. Un data provider è dunque uno strato di software che permette ad un'applicazione di interagire con dei database, strato che costituisce un'interfaccia comune per ricavare dei dati da utilizzare poi per diversi scopi, ad esempio per visualizzare dei report, o per fare dei calcoli su di essi. Questa interfaccia è comune a tutti i provider, dunque, qualunque sia il database sottostante, è possibile utilizzare le stesse procedure per lavorare con essi. Tutto ciò è possibile grazie all'architettura ad interfacce e classi astratte, con cui tutti i provider sono implementati. Il bello dell'architettura è che non è limitata all'accesso ad un database commerciale, ma può essere sfruttata anche per accedere a fonti di dati personalizzate, ad esempio ad un database personale, al registro di sistema, o ai tipi di un assembly, o a qualunque tipo di dato si possa immaginare. In questo articolo vedremo quali sono le classi e le interfacce che un custom provider deve esporre, e le implementeremo per creare un minimo provider di accesso al file system, per effettuare delle ricerche come se si trattasse di una sorta di database SQL.

L'ARCHITETTURA DI UN PROVIDER

Un .NET Data Provider è costituito in genere da quattro classi principali, che implementano altrettante interfacce e che collaborano fra di esse in maniera praticamente standard.
La classe Connection è una classe richiesta anche se il provider non dovrà connettersi ad un database relazionale. Le altre classi utilizzano una Connection per connettersi a qualsiasi genere di dati. Ad esempio i DataAdapter usano una connessione per riempire un DataSet. La classe Command rappresenta un comando da inviare ad una sorgente dati, per un database, ad esempio, un Command sarà un classico Insert, Update e così via, oppure Select se è un comando che restituisce dei record.
La classe DataReader si occupa di processare i risultati di un Command in maniera sequenziale, partendo dal primo fino all'ultimo in maniera forward-only. Inoltre per ogni riga consente di leggere diversi attributi, ad esempio per un database consentirà di leggere i vari campi di un record.
La classe DataAdapter riempie un DataSet con i risultati di un Command, ed è utilizzata per rimandare indietro alla sorgente dati, aggiornandola, le eventuali modifiche fatte sul DataSet.
Tali classi implementano delle interfacce anch'esse naturalmente standard, dunque avremo un'interfaccia IDbConnection, una IDbCommand, la IDbDataAdapter, e la IDataReader. Esistono poi altre classi a contorno, più specialistiche, che servono ad esempio per aggiungere il supporto alle transazioni, dei parametri ai comandi, o anche delle classi di eccezioni personalizzate,

SCRIVERE UN PROVIDER

Per mostrare l'implementazione di un custom provider, che svolga dei compiti di ricerca sul file system, creeremo alcune delle classi standard viste, in particolare una che rappresenti la connessione ad una directory dell'hard disk, una per i comandi di ricerca da inviare, ed un DataReader per leggere sequenzialmente i risultati. Non ha senso in questo caso un DataAdapter, in quanto vogliamo solo ricavare dei dati in lettura.
Non tutti i metodi e le proprietà delle interfacce saranno implementate, in quanto non avrebbero alcune volte utilità, ed in quanto non abbiamo lo scopo di creare un provider ADO.NET completo sotto tutti i punti di vista. Chi vorrà, comunque,

Conoscenze richieste
conoscenze medie di C#

Software
NET Framework 2.0, Microsoft Visual Studio .NET 2005

Impegno

Tempo di realizzazione

potrà estendere il provider, con nuovi comandi ad esempio, o implementando le funzioni assenti.

IL FILESYSTEMPROVIDER

Il provider che verrà implementato nel corso dell'articolo ha lo scopo di consentire la ricerca di file e directory sul file system, a partire da una certa directory di partenza, utilizzando dei comandi simil-SQL.
Ad esempio per farsi restituire tutte le sotto-directory di una data cartella, insieme ad informazioni, come la data di creazione ed il numero di file contenuti in esse potremo utilizzare un comando simile a "SELECT dir", per ottenere invece solo i file scriveremo "SELECT files", o "SELECT *" per entrambi i tipi. Inoltre implementeremo anche una sorta di filtro, per ricercare solo determinati file o cartelle, con comandi ad esempio del tipo "SELECT file WHERE Name LIKE 'a*'" per ottenere i file che iniziano per 'a'.
La figura seguente mostra il diagramma delle tre classi che si andrà ad implementare.

LA CLASSE FILESYSTEMCONNECTION

La prima classe da creare è la FileSystemConnection, che implementerà l'interfaccia IDBConnection. Sebbene l'esempio in questione non abbia bisogno di un concetto analogo a quello di connessione ad un database, è comunque necessario implementare la classe, in quanto le altre, ad esempio Command, utilizzano le funzionalità di una IDbConnection.

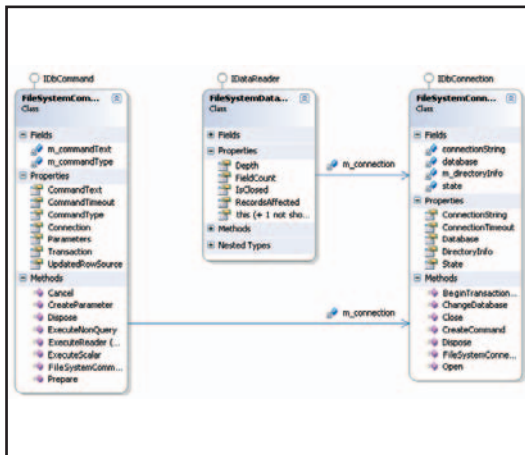
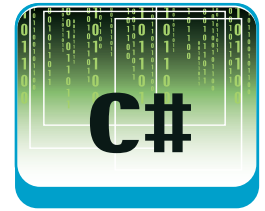
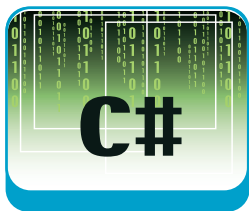



Fig. 1: diagramma delle classi del FileSystemProvider

La classe FileSystemConnection è mostrata qui di seguito. Alcuni metodi di IDBConnection sono dei semplici stub vuoti che generano un'eccezione NotSupportedException nei casi in cui il metodo non è implementato, perché nel caso di una "connessione" al file system, non servirebbe.
Una connessione utilizza normalmente una stringa per conoscere i parametri con cui connettersi ad un database, in questo caso l'unico parametro è il percorso della directory su file system, dal quale iniziare la ricerca. Quindi la stringa di connessione per una FileSystemConnection sarà ad esempio scritta nel formato:

```
path="C:\"
```

Da tale stringa la classe ricaverà il percorso, memorizzandolo all'interno del campo database.

```
public class FileSystemConnection:IDbConnection
{
    private DirectoryInfo m_directoryInfo;
    private string database = "";
    private string connectionString = "";
    private ConnectionState state =
                                    ConnectionState.Closed;
    public FileSystemConnection()
    {
    }
    public FileSystemConnection(string connString)
    {
      ConnectionString = connString;
    }
    #region IDbConnection Members
    public IDbTransaction BeginTransaction(Isolatio
                                                    Level il)
    {  throw new NotSupportedException("The
            method or operation is not implemented.");   }
    public IDbTransaction BeginTransaction()
    {   throw new NotSupportedException("The
            method or operation is not implemented.");   }
    public void ChangeDatabase
                                    (string databaseName)
    {
      database = databaseName;
    }
    public void Close()
    {
      database = null;
      m_directoryInfo = null;
    }
    public DirectoryInfo DirectoryInfo
    {
      get
      {
        return m_directoryInfo;
      }
    }
        public string ConnectionString
    {
      get
      {
        return connectionString;
      }
      set
      {
```

```
            connectionString=value;
            if (connectionString.StartsWith("path="))
                database = connectionString.
                                        Substring(5);
        }
    }
    public int ConnectionTimeout
    {
        get { return 0; }
    }
    public IDbCommand CreateCommand()
    {
        IDbCommand command =
            (IDbCommand)(new FileSystemCommand());
        command.Connection = this;
        return command;
    }
    public string Database
    {
        get { return database; }
    }

    public void Open()
    {
        if (connectionString == null || connectio
                              String.Length == 0)
        {
            throw new ArgumentException("Connection
                        string non inizializzata");
        }
        if (Directory.Exists(database))
        {
            m_directoryInfo = new DirectoryInfo
                                       (database);
            this.state = ConnectionState.Open;
        }
        else
        {
            throw new FileSystemException
("Directory '" + m_directoryInfo + "' not found.");
        }
    }
    public ConnectionState State
    {
        get
        {
            return state;
        }
    }
    #endregion
    #region IDisposable Members
    public void Dispose()
    {
        if(this.state==ConnectionState.Open)
            Close();
        connectionString = null;
    }
    #endregion
}
```

Il primo metodo invocato su di una Connection dopo la sua costruzione, è il metodo Open, che serve nel nostro caso a ricavare le informazioni sul "database" tramite un oggetto DirectoryInfo, naturalmente dopo aver verificato che il percorso indicato dalla stringa di connessione sia valido ed esistente.
Nel caso in cui la ConnectionString sia valida, e dunque l'apertura può andare a buon fine, il ConnectionState sarà impostato ad Open.
Il metodo CreateCommand inoltre permette di creare un FileSystemCommand, illustrato nel prossimo paragrafo, a partire da una FileSystemConnection.

IMPARTIRE COMANDI CON FILESYSTEMCOMMAND

La classe FileSystemCommand deriva da IDbCommand, e rappresenta dunque un comando da impartire in genere ad un database, ed in questo caso dunque un comando di ricerca su file system.
La struttura generale è la seguente:

```
public class FileSystemCommand:IDbCommand
{
    private FileSystemConnection m_connection;
    private string m_commandText;
    private CommandType m_commandType;
        ...
}
```

Si hanno dunque una FileSystemConnection, creata ed aperta come nel precedente paragrafo, il comando vero e proprio, ed il tipo di comando, che in questo esempio sarà sempre un CommandType.Text, in quanto non sono previste tabelle o stored procedure. Un tentativo di usare tali tipi genererà un'eccezione:

```
public CommandType CommandType
{
        get
        {
            return m_commandType;
        }
        set
        {
            if (value != CommandType.Text)
                throw new
ArgumentException("L'argomento non è valido.","value");
            m_commandType = value;
        }
}
```

La modalità di lettura dei risultati di un comando è quella sequenziale, realizzata tramite un FileSystemReader. Per ottenere un simile oggetto viene invocato

il metodo ExecuteReader, che verifica l'apertura della connessione e quindi costruisce un FileSystemReader.

```
public IDataReader ExecuteReader()
{
// Verifica che la connessione esiste ed è aperta
        if (m_connection == null )
        {
            throw new ArgumentNullException
                        ("La connessione non esiste");
        }
        else if( m_connection.State !=
                        ConnectionState.Open)
            throw new InvalidOperationException
            ("La connessione deve essere prima aperta");
        FileSystemDataReader reader = new
            FileSystemDataReader((FileSystemConnection)
                            m_connection, this);
        return reader;
}
```

LEGGERE I DATI CON FILESYSTEMREADER

La classe FileSystemReader costituisce il cuore del provider, in quanto implementa le parti sostanziali che effettuano la ricerca e che permettono di scorrere e leggere i risultati ottenuti. Si vuole dare la possibilità di ricercare file, directory, o entrambi contemporaneamente. Dunque si avrà bisogno di tre differenti campi in cui si manterranno i risultati:

```
private FileInfo[] m_files;
private DirectoryInfo[] m_directories;
private FileSystemInfo[] m_fileSystemInfos;
```

La modalità di lettura sarà determinata e mantenuta da un campo del tipo enumerativo ReadMode:

```
private enum ReadMode
{
Directory,
File,
All
}
private ReadMode m_readMode;
```

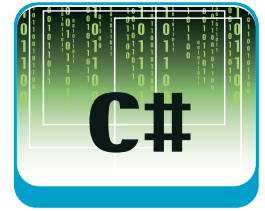
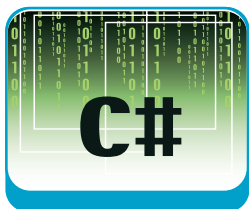
Ogni tipologia, inoltre, avrà differenti attributi leggibili, che vengono definiti in tre array di stringhe corrispondenti alle tipologie di oggetti:

```
private String[] m_dirFields = { "Type" ,"Name",
                        "Files", "CreationTime"};
private String[] m_fileFields = { "Type", "Name",
                        "Size", "CreationTime" };
private string[] m_allFields= {"Type", "Name",
                        "DirSize_Or_Files#", "CreationTime" };
```

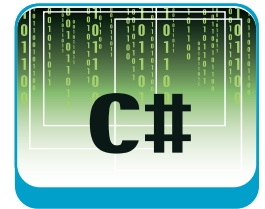
Altri campi poi serviranno ad esempio a mantenere il numero di record ottenuti, il puntatore al record attuale quando si procederà alla lettura sequenziale, ed il record stesso, che data la natura eterogenea di essi, sarà un array di object:

```
private object[] m_row = new object[4];
private int m_cursor = 0;
private int m_recordNumber;
```

Il costruttore della classe FileSystemReader, prendendo in ingresso una FileSystemConnection ed un FileSystemCommand

```
public FileSystemDataReader(FileSystemConnection
            connection,FileSystemCommand command)
{
        m_connection = connection;
        m_startDirectory = connection.DirectoryInfo;
        m_cursor = 0;
        string cmdLower =
                command.CommandText.ToLower();
// cosa ricercare?
        if ( cmdLower== "select *")
        {
            m_readMode = ReadMode.All;
            m_fileSystemInfos = m_startDirectory.Ge
                                FileSystemInfos();
            m_recordNumber =
                        m_fileSystemInfos.Length;
        }
        else if (cmdLower == "select files")
        {
            m_readMode = ReadMode.File;
            m_files = m_startDirectory.GetFiles();
            m_recordNumber = m_files.Length;
        }
        else if (cmdLower == "select dir")
        {
            m_readMode = ReadMode.Directory;
            m_directories = m_startDirectory.
                                GetDirectories();
            m_recordNumber = m_directories.Length;
        }
        else if (cmdLower.StartsWith("select *")
            && cmdLower.IndexOf(" where name like
                                        ")>7)
        {
            string filter=cmdLower.Substring(cm
                        Lower.IndexOf("'")+1);
            if(!filter.EndsWith("'"))
                throw new ArgumentException("Errore di
        sintassi nel filtro del comando \"" + cmdLower+"\"");
            filter=filter.Substring(0,filter.Length-1);
            m_readMode = ReadMode.All;
            m_fileSystemInfos = m_startDirectory.Ge
```

```
                                    FileSystemInfos(filter);
            m_recordNumber = m_fileSystemInfos.Length;
        }
        else if (command.CommandText.ToLower() ==
                                            "select files")
        {
            m_readMode = ReadMode.File;
            m_files = m_startDirectory.GetFiles();
            m_recordNumber = m_files.Length;
        }
        else if (command.CommandText.ToLower() ==
                                              "select dir")
        {
            m_readMode = ReadMode.Directory;
            m_directories = m_startDirectory.
                                        GetDirectories();
            m_recordNumber = m_directories.Length;
        }
        else
        {
            throw new ArgumentException("Errore di
             sintassi nel comando \"" + command.Command
                                              Text+"\"");
        }
        m_isClosed = false;
    }
```

Il comando da eseguire è contenuto nella proprietà CommandText dell'oggetto command, ed è dunque con uno switch che si può distinguere quale tipo di oggetti ricercare ed interpretare il comando stesso ed i suoi eventuali filtri. Il simil-sql utilizzato in questo esempio permette solo operazioni di ricerca, quindi si è utilizzata una keyword select, con la possibilità di ricercare solo file, con select files, oppure solo directory con select dir, o entrambi con un select *.
Inoltre sarà possibile aggiungere un filtro 'where name like' seguito dalle classiche wildcard valide per i file system. La lettura dei risultati avverrà poi invocando sequenzialmente il comando Read, per verificare che siano ancora presenti record da leggere e per impostare il campo m_row con i dati del record attuale, cioè creando leggendo ad esempio il nome, la data di creazione, la dimensione di un file, oppure, per una directory, nome, data di creazione e file contenuti in essa.

```
public bool Read()
{
  bool notLast;
  try
  {
     switch (m_readMode)
     {
            case ReadMode.All:
  notLast= m_cursor < m_fileSystemInfos.Length;
                 if (notLast)
              {
                    m_row[1] =
                            m_fileSystemInfos[m_cursor].Name;
                    m_row[3] =
                   m_fileSystemInfos[m_cursor].CreationTime;
                    if (Directory.Exists(m_fileSyste
                                  mInfos[m_cursor].FullName))
                    {
                       m_row[0] = "dir";
                       m_row[2] = new DirectoryInfo(m_file
      SystemInfos[m_cursor].FullName).GetFiles().Length;
                    }
                    else
                    {
                       m_row[0] = "file";
                       m_row[2] = new FileInfo
         (m_fileSystemInfos[m_cursor].FullName).Length;
                    }
                    m_cursor++;
                    return true;
              }
              return false;
              case ReadMode.File:
              notLast = m_cursor < m_files.Length;
              if (notLast)
              {
                   m_row[0] = "file";
                   m_row[1] = m_files[m_cursor].Name;
                   m_row[2] = m_files[m_cursor].Length;
                   m_row[3] = m_files[m_cursor].
                                              CreationTime;
                   m_cursor++;
                   return true;
               }
               return false;
              case ReadMode.Directory:
              notLast = m_cursor <
                                 m_fileSystemInfos.Length;
              if (notLast)
              {
                  m_row[0] = "dir";
                  m_row[1] =
                         m_directories[m_cursor].Name;
                  m_row[2] =
                m_directories[m_cursor].GetFiles().Length;
                  m_row[3] = m_fileSystemInfos
                                   [m_cursor].CreationTime;
                  m_cursor++;
                  return true;
               }
               return false;
           }
           return false;
     }
     catch (UnauthorizedAccessException uex)
     {
          m_row[2] = "Accesso negato";
          m_cursor++;
          return true;
```

```
    }
    catch(Exception ex)
    {
        throw new FileSystemException("Errore
            di sintassi nella query o directory non trovata");
    }}
```

Le classi create interagiscono strettamente fra di loro, come mostrato nel sequence diagram della seguente figura che riassume quanto verrà realizzato nell'applicazione di esempio.

UN'APPLICAZIONE DI PROVA

Con le classi realizzate, quindi con il nuovo FileSystemProvider si può ora realizzare un'applicazione che effettua delle ricerche in una data directory del file system. L'interfaccia dell'applicazione è mostrata in figura 3, e naturalmente tralasciamo la realizzazione dei controlli e delle finestre, rinviando al codice contenuto nel cd in allegato. La parte interessante avviene al clic sul pulsante cerca:

```
private void btQuery_Click(object sender, EventArgse)
{
        string connString = "path=" +
                            txtDirectory.Text;
    try
    {
        using (FileSystemConnection conn = new
                    FileSystemConnection(connString))
        {
            conn.Open();
            FileSystemCommand cmd =
        (FileSystemCommand) conn.CreateCommand();
            cmd.CommandText=txtQuery.Text;
            FileSystemDataReader dr=
        (FileSystemDataReader) cmd.ExecuteReader();
            PrepareListView(dr);
            ListViewItem item;
```

Fig. 2: *L'applicazione di ricerca in azione*

```
            listView1.Items.Clear();
            errorProvider.SetError(txtQuery, "");
            errorProvider.SetError(txtDirectory, "");
            while(dr.Read())
            {
                item = new ListViewItem(dr[0].ToString());
                if (item.Text == "dir")
                    item.ImageIndex = 0;
                else if (item.Text == "file")
                    item.ImageIndex = 1;
                item.SubItems.Add(dr[1].ToString());
                item.SubItems.Add(dr[2].ToString());
                item.SubItems.Add(dr[3].ToString());
                listView1.Items.Add(item);
            }}}
```

Fig. 3: *Sequence diagram di una ricerca*

```
        catch (Exception ex){
            MessageBox.Show(ex.Message);
        }}
```

Chi ha già avuto a che fare con i data provider ADO.NET per l'accesso ai database noterà la perfetta analogia dei concetti e dell'utilizzo delle classi. Basta creare una connessione, aprirla, quindi creare un comando, con una sorta di linguaggio simil-sql, ad esempio un select dir o select files, ed eseguirlo per ottenere un DataReader. Infine utilizzare quest'ultimo per scorrere i risultati ed aggiungerli ad una ListView.

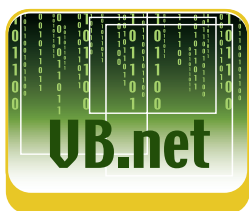
CONCLUSIONI

Abbiamo visto come realizzare un DataProvider personalizzato utilizzando le interfacce messe a disposizione dal .NET Framework, ed i concetti classici dei provider per l'accesso ai database di ADO.NET . Non vi resta che cercare di migliorarlo aggiungendo le caratteristiche non implementate, oppure creare il provider che può servire a risolvere un vostro problema

Antonio Pelleriti

Potete rivolgere domande di chiarimenti o ulteriori richieste all'autore all'indirizzo antonio.pelleriti@ioprogrammo.it, o ancora sul forum di IoProgrammo (http://forum.ioprogrammo.net) o sul sito www.dotnetarchitects.it.

METTI UN FTP NEL TUO BROWSER

CREIAMO UN CLIENT FTP CHE VIVE ALL'INTERNO DI EXPLORER. UTILISSIMO PER UPLOADARE FILE IN REMOTO SENZA SCOMODARE PROGRAMMI DI TERZE PARTI. IMPAREREMO COME VB.NET GESTISCE ALCUNE FUNZIONI RELATIVE AL NETWORKING

Nella versione 1.0 di .NET, una delle cose inspiegabilmente mancanti era il supporto al protocollo FTP (File Transport Protocol).

Esisteva la possibilità di ricorrere, è vero, a librerie (anche open source) prodotte da terze parti. Tuttavia il supporto FTP introdotto da .NET 2.0 è sicuramente una buona notizia.

Il meccanismo di funzionamento dell'API FTP di .NET 2.0 è piuttosto semplice, il workflow dell'operazione è :

1. Instanziare un oggetto FtpWebRequest impostandone modalità di connessione e url
2. Impostare il comando FTP attraverso la proprietà method dell'oggetto FtpWebRequest
3. Eventualmente scrivere sul flusso dei dati dal client al server recuperando lo Stream in uscita attraverso la funzione GetRequestStream dell'oggetto FtpWebRequest (questa operazione serve solo per l'upload dei file verso il server)
4. Recuperare la risposta del server attraverso la funzione GetResponse dell'oggetto FtpWeb Request che restituisce un oggetto di tipo FtpWebResponse e, attraverso questo, recuperare eventualmente anche il flusso di dati dal server verso il client con il metodo GetResponseStream.

Ad esempio, per scaricare un file da un server FTP è sufficiente un codice di questo tipo:

```
Public Sub DownloadFile(ByVal url As Uri, ByVal
        localFilename as String, ByVal credentials As
                                    NetworkCredential)
    Dim ftpRequest As FtpWebRequest =
                          FtpWebRequest.Create(Url)
    ftpRequest.Credentials = credentials
    ftpRequest.Method =
              WebRequestMethods.Ftp.DownloadFile
    ftpRequest.UseBinary = True
    ftpRequest.KeepAlive = False
    Dim ftpResponse As FtpWebResponse =
                          ftpRequest.GetResponse
  Dim reader As New
```

```
  IO.StreamReader(ftpResponse.GetResponseStream)
    Dim writer As New
                    IO.StreamWriter(localFilename)
    writer.Write(reader.ReadToEnd())
    reader.Close()
    writer.Close()
  ftpResponse.Close()
End Sub
```

Recentemente mi è capitato di utilizzare questa tecnologia per applicazioni web. Il problema è quello, abbastanza diffuso tra gli sviluppatori, della pubblicazione di applicazioni web presso provider esterni, dove non si ha il controllo sulle permissions assegnate alle directory del file system. Il provider, infatti, prevede soltanto una directory nella quale l'utente web aveva anche diritti di scrittura: la directory /public del sito.

Mi sono trovato ad installare applicazioni, sviluppate da terzi, che invece richiedevano percorsi di scrittura diversi, purtroppo non parametrizzati.

La prospettiva era quindi quella di rivedere tutto il codice per cambiare i percorsi di scrittura.

Grazie all'API FTP di .NET 2.0 è invece stato possibile scrivere non attraverso il file system, ma attraverso l'account FTP che il provider metteva a disposizione.

Sulla scorta di questa esperienza mi è allora venuta l'idea di sviluppare un vero e proprio client FTP, da utilizzare via web, per esplorare fino in fondo le possibilità offerte dall'API.

FTP WEB CLIENT

L'applicazione, per adesso sviluppata al solo scopo sperimentale, si propone di :

- Consentire il browsing delle cartelle di un server FTP
- Consentire la cancellazione, il download e l'upload di files

Il tutto non attraverso una Windows Application, ma in ASP.NET.

PANORAMICA SULL'APPLICAZIONE

Prima di vedere nel dettaglio alcuni particolari dell'implementazione, analizziamo il funzionamento dell'applicazione finita.
L'applicazione si installa semplicemente copiando la directory che la contiene sul disco fisso e configurando, nel server web IIS, una directory virtuale in corrispondenza di tale directory.
Poniamo quindi che in IIS sia stata configurata la directory virtuale con il nome ftpclient, aprite il vostro browser all'indirizzo http://localhost/ftpclient ed apparirà la maschera di collegamento al server FTP di cui in figura 1.

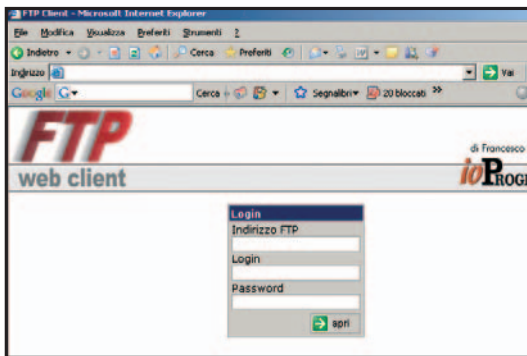



Fig. 1: *La maschera di login iniziale*

La maschera di login chiede la url del server FTP, il nome utente e la password per la connessione.
Una volta fornite queste informazioni, la web application presenta una vista sulla directory principale (root) del server FTP (figura 2).

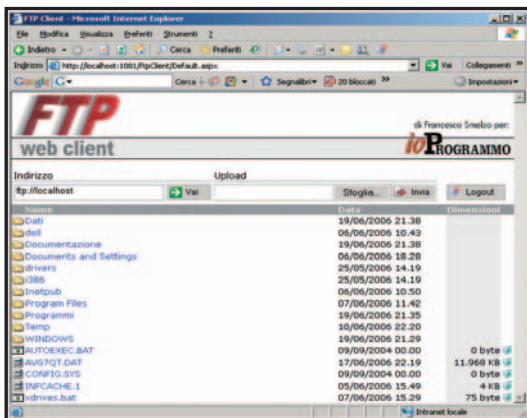
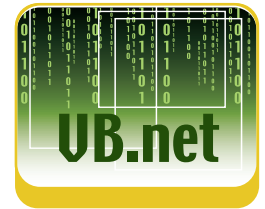



Fig. 2: *Il browsing FTP da Web*

Per effettuare le prove abbiamo qui configurato il servizio FTP di IIS puntandolo direttamente sul disco C: della macchina che quindi risponde all'url ftp://localhost .
Cliccando sulle cartelle si apre la directory inferiore, mentre cliccando sui files si procede al download.
Da notare le icone delle directory e dei files, le stesse del sistema operativo, vedremo successivamente come ottenerle.

Sul resto non c'è molto altro da dire: c'è un campo upload per i files da caricare, un link per cancellare il file, una barra degli indirizzi e un bottone per il logout. Insomma, seppur spartano, un client FTP vero e proprio ma … nel browser!
Ma andiamo ad analizzare adesso la struttura ed i passi salienti dell'applicazione. Non ci soffermeremo molto sui particolari dell'interfaccia (che nel codice è stata mantenuta volutamente semplice per rendere più chiara la logica applicativa).

SICUREZZA

L'applicazione Web illustrata in questo articolo e allegata al CD è perfettamente funzionante e può essere utilizzata liberamente. Attenzione però a pubblicarla in un sito Web in modalità non protetta da password: gli utenti anonimi del sito potrebbero compiere qualsiasi operazione (cancellazione o upload) nei confronti di server FTP di cui conoscessero le credenziali di autenticazione. Queste operazioni verrebbero però registrate dal server FTP come provenienti dal vostro sito web e quindi voi ne sareste responsabili!
È pertanto consigliabile utilizzare l'applicazione soltanto in aree riservate del sito o in intranet, oppure modificare l'applicazione aggiungendo un proprio logging delle operazioni compiute dagli utenti.
Per consentire una prova "sul campo" l'applicazione è stata pubblicata all'indirizzo http://www.smelzo.it/ftp in modalità "demo" ovvero senza la possibilità di effettuare la cancellazione e la modifica di files.

IL PROGETTO

Per lo sviluppo del nostro mini-progetto abbiamo utilizzato Visual Studio 2005 nel quale abbiamo impostato un nuovo progetto web come vediamo in figura 3.

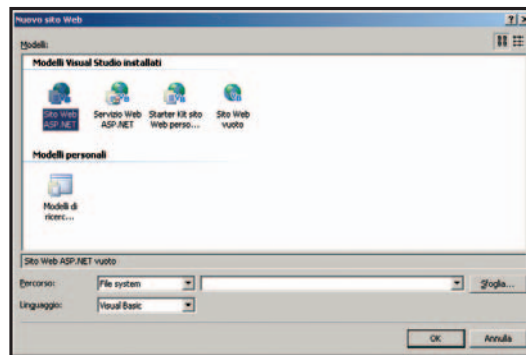



Fig. 3: *Un nuovo progetto Web in Visual Studio 2005*

Lo sviluppo web in Visual Studio 2005 ha fatto veramente dei passi avanti notevoli: adesso è infatti possibile inserire le nostre classi nella cartella App_Code del progetto ed è possibile eseguire il debug senza dover ricompilare ogni volta. Inoltre, nelle pagine aspx, oltre al modello code-behind (dove il codice risiede in file separati) è adesso pos-

sibile finalmente utilizzare il modello online dove, come in ASP classico, possiamo scrivere il codice direttamente nella pagina (questa non è però la modalità predefinita: per attivarla quando inseriamo nel progetto una nuova pagina aspx dobbiamo deselezionare la casella di spunta “Inserisci codice in file separato” come mostrato in figura 4). Naturalmente potevamo utilizzare il modello inline anche nelle versioni precedenti di Visual Studio, soltanto che adesso abbiamo il pieno supporto Intellisense anche all’interno del tag <script> che ospitano il codice lato server (vedi figura 5).

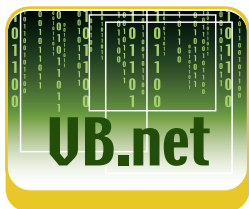
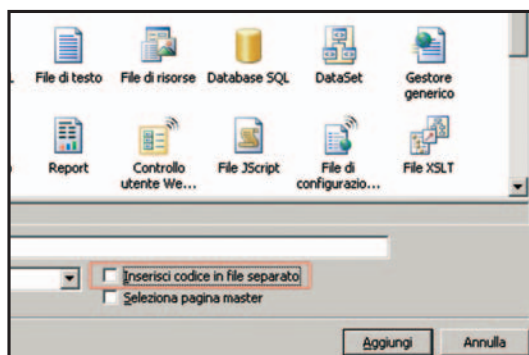



Fig. 4: *Utilizzo del modello inline per le pagine ASP.NET*

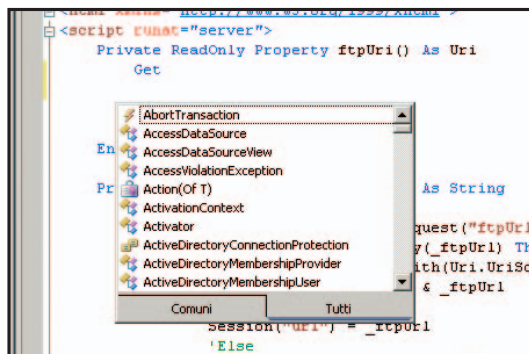



Fig. 5: *Supporto intellisense al modello inline*

Naturalmente anche noi abbiamo a cuore il paradigma della separazione e della riutilizzabilità del codice, solo che ci sembra spesso esagerato e dispersivo ricorrere allo sviluppo di librerie esterne per progetti di piccole e medie dimensioni. Quindi abbiamo optato per l’inserimento della nostra logica applicativa in file di classi nella cartella App_Code (classi che tra l’altro possono essere tranquillamente riutilizzate in progetti diversi) e l’utilizzo dei file aspx per la sola logica di presentazione.
Per prima cosa abbiamo quindi inserito nel progetto, nella cartella App_Code, il file di codice contenente le classi deputate a gestire il colloquio FTP. In questo file abbiamo creato la classe di base, chiamata FtpClient, che verrà utilizzata dal lato ASP.NET per colloquiare con il server FTP.
Come possiamo vedere dalla figura 6, nella classe FtpClient sono state dichiarate quattro variabili a livello di classe:

- Url – di tipo Uri che rappresenta l’url della richiesta FTP
- Credentials – di tipo NetworkCredential che rappresenta le credenziali di autenticazione presso il server
- Proxy – di tipo IWebProxy che rappresenta l’eventuale proxy da utilizzare nella connessione
- Port – di tipo Integer che rappresenta la porta TCP per la connessione (normalmente la 21)

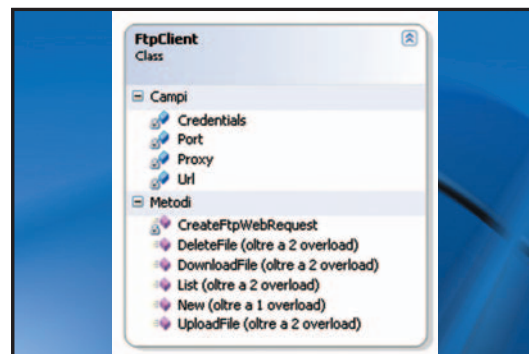



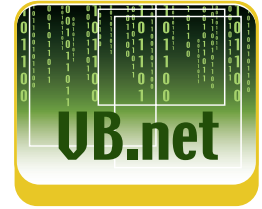
Fig. 6: *Struttura della classe FtpClient*

Tali variabili vengono valorizzate nei costruttori di cui vengono fornite due versioni:

```
Sub New(ByVal url As Uri, ByVal credentials As
        System.Net.NetworkCredential, ByVal proxy As
                IWebProxy, ByVal port As Integer)
    Me.Url = url
    Me.Port = port
    If url.Port <> port Then
        'aggiunge la porta se mancante nella url
        Dim builder As New UriBuilder(url)
        builder.Port = port
        Me.Url = builder.Uri
    End If
    Me.Credentials = credentials
    Me.Proxy = proxy
End Sub

Sub New(ByVal url As Uri, ByVal userName As String,
      ByVal password As String, ByVal port As Integer)
    Me.New(url, New NetworkCredential(userName,
                      password), Nothing, port)
End Sub
```

Dove nel secondo costruttore, più semplice e di uso più comune, si omette il proxy e si costruisce direttamente la credenziale a partire da userName e password.

È stata poi aggiunta una funzione, di uso comune a tutti i metodi, per creare la richiesta al server FTP:

```
Private Function CreateFtpWebRequest(ByVal method
      As String, Optional ByVal useBinary As Boolean =
```

```
                                    True) As FtpWebRequest
        Dim ftpRequest As FtpWebRequest =
                                 FtpWebRequest.Create(Url)
        ftpRequest.Credentials = Me.Credentials
        ftpRequest.KeepAlive = False
        ftpRequest.UseBinary = useBinary
        ftpRequest.Method = method
        If Me.Proxy IsNot Nothing Then
            ftpRequest.Proxy = Me.Proxy
        End If
        Return ftpRequest
End Function
```

È da sottolineare che il parametro method, rappresenta il comando FTP da inviare al server. Non è necessario scrivere manualmente la stringa del comando in quanto è possibile utilizzare le costanti esposte dalla classe System.Net.Web RequestMethods.Ftp che sono autodescrittive.

I metodi pubblici esposti dalla classe sono:
DeleteFile – che invia un comando di cancellazione del file
DownloadFile – che scarica un file dal server come flusso
List – che ottiene una lista di oggetti contenuti in una directory del server
UploadFile – che invia un file al server come matrice di byte

I comandi gestiti dalla classe sono solo una parte di quelli disponibili; mancano, ad esempio, quelli per rinominare, creare e cancellare le directory, ma per adesso non si è voluto complicare troppo le cose e limitarsi solo all'essenziale.

IL PARSING DEL LIST

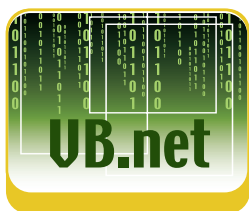
Dei quattro metodi citati sicuramente il più complesso è il List. Di base l'API FTP dispone di due comandi per ottenere la lista di oggetti di una directory nel server : WebRequestMethods.Ftp.List Directory e WebRequestMethods.Ftp.ListDirectory Details.
Il primo comando è inadatto ai nostri scopi perché fornisce soltanto il nome degli oggetti senza dirci, ad esempio, se si tratta di file o directory.
Il secondo invece è più completo perché fornisce la lista comprendente informazioni su: natura dell'oggetto(file o directory), dimensioni e data di creazione.
Il problema è che la struttura della lista dipende dal server, nei sistemi Unix e Linux la risposta sarà del tipo illustrato in figura 7, mentre per i server Windows basati su IIS la risposta sarà quella mostrata in figura 8.

Fig. 7: ***Listing su server FTP Unix/Linux***

Fig. 8: ***Listing su server FTP IIS***

Abbiamo quindi dovuto creare una classe astratta, FtpListParserBase, che poi verrà ereditata dalle classi deputate ad interpretare la risposta del server (noi abbiamo implementato soltanto FtpListParserIIS e FtpListParser Unix) ognuna di queste sottoclassi deve avere un metodo pubblico, Match,che, partendo dalla prima linea di testo ricevuta dal server, determina se è essa a poter effettuare il parsing oppure no.

Fig. 9: ***Classi per il parsing del listing***

Senza scendere qui troppo in dettaglio sul parsing (effettuato con le Regular Expressions) basti dire che le classi specializzate di FtpListParserBase si comportano tutte nello stesso modo prendono una matrice di linee di testo, ne interpretano il contenuto e creano una lista corrispondente di oggetti. Questi oggetti sono basati su un'altra classe, FtpListItem, che raccoglie le informazioni dalla linea di testo inviata dal server dimensioni, data, nome ecc…

COSA VUOL DIRE FTP?

L'FTP, acronimo di File Transfer Protocol (protocollo di trasferimento file), è un servizio che garantisce il trasferimento di file tra client e server. Il protocollo FTP che ha subito una lunga evoluzione negli anni (il primo meccanismo di trasmissione file risale al 1971, fu sviluppato presso il MIT) è ancora largamente utilizzato soprattutto dagli Host Providers per garantire ai clienti l'accesso al proprio sito web.

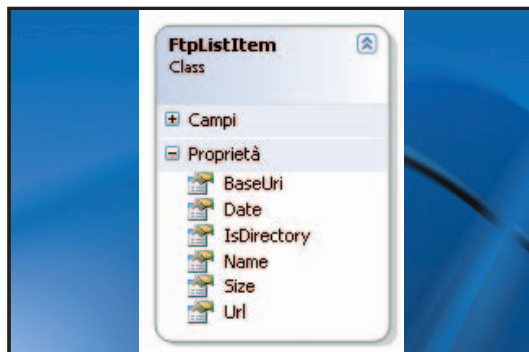



Fig. 10: Classi FtpListItem

Tutti gli oggetti FtpListItem vengono raccolti dal parser in un insieme FtpList che dispone di metodi per l'ordinamento e la contestualizzazione degli elementi contenuti.

Il nostro metodo List della classe FtpClient si presenterà quindi, alla fine, così:

```
Public Function List() As FtpList
    Dim ftpRequest As FtpWebRequest = _
Me.CreateFtpWebRequest(WebRequestMethods.Ftp.ListDirectoryDetails, False)
    Dim ftpResponse As FtpWebResponse =
                ftpRequest.GetResponse
    Dim stream As IO.Stream =
                ftpResponse.GetResponseStream
    Dim reader As New IO.StreamReader(stream)
    Dim lines As New List(Of String)
    While Not reader.Peek = -1
        lines.Add(reader.ReadLine)
    End While
    If lines.Count = 0 Then
        ftpResponse.Close()
        Return FtpList.Empty(Url)
    End If
    ftpResponse.Close()
    Dim parser As FtpListParserBase
    If FtpListParserUnix.Match(lines(0)) Then
        parser = New FtpListParserUnix(Me.Url)
    ElseIf FtpListParserIIS.Match(lines(0)) Then
        parser = New FtpListParserIIS(Me.Url)
    Else
        parser = Nothing
    End If
    If parser Is Nothing Then
        Throw New WebException("Linea non
                riconosciuta ")
    End If
    Return parser.Parse(lines.ToArray)
End Function
```

Dove si leggono le linee trasmesse dal server, si sceglie il parser adatto e si utilizza per restituire un insieme di oggetti FtpListItem.

Gli altri metodi esposti dalla classe FtpClient sono invece più semplici.

CANCELLAZIONE DI FILE

Il codice per la cancellazione di un file dal server FTP sarà:

```
Public Function DeleteFile() As String
 Dim ftpRequest As FtpWebRequest = _
Me.CreateFtpWebRequest(WebRequestMethods.Ftp.DeleteFile)
    Dim ftpResponse As FtpWebResponse =
                ftpRequest.GetResponse
    Dim r As String =
                ftpResponse.StatusDescription
    ftpResponse.Close()
    Return r
End Function
```

Il valore di ritorno sarà, in questo caso la stringa contenente il messaggio di ritorno del server.

DOWNLOAD DI FILE

La funzione di download, invece, restituisce un flusso (stream) dei dati provenienti dal server FTP :

```
Public Function DownloadFile() As IO.Stream
    Dim ftpRequest As FtpWebRequest =
Me.CreateFtpWebRequest(WebRequestMethods.Ftp.DownloadFile)
    Dim ftpResponse As FtpWebResponse =
                ftpRequest.GetResponse
    Return ftpResponse.GetResponseStream()
End Function
```

UPLOAD DI FILE

Conclude la serie di funzioni esposte dalla classe FtpClient il metodo UploadFile:

```
  Public Function UploadFile(ByVal b() As Byte) As
                String
    Dim ftpRequest As FtpWebRequest =
Me.CreateFtpWebRequest(WebRequestMethods.Ftp.UploadFile)
    ftpRequest.ContentLength = b.Length
    Dim requestStream As IO.Stream =
                ftpRequest.GetRequestStream
    requestStream.Write(b, 0, b.Length)
    requestStream.Close()
    Dim ftpResponse As FtpWebResponse =
                ftpRequest.GetResponse
    Dim r As String =
                ftpResponse.StatusDescription
    ftpResponse.Close()
    Return r
  End Function
```

Anch'esso, come deleteFile, restituisce la stringa contenente il messaggio di ritorno del server, tuttavia richiede, come parametro la matrice di byte che rappresenta il file da caricare.

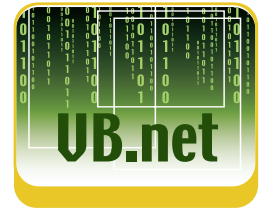
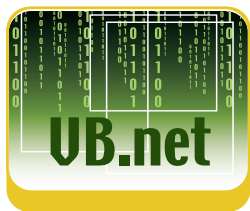
IL LATO WEB

Sul versante web, la nostra pagina ASP.NET non farà altro che utilizzare la classe FTP client definita in App_Code.
Per l'implementazione non è stato utilizzato il modello dei controlli ASP.NET collegato agli Eventi (Textbox, Button ecc…), perché personalmente ritengo che non faccia altro che complicare la vita… Preferisco invece attenermi al flusso Request➠Response che è sempre alla base del ciclo della pagina. Su OnLoad della pagina, quindi, si innesta tutto il ciclo delle operazioni.
Si controlla se la Request o la Session contengono i valori di autenticazione, se non li contengono verrà mostrata la maschera di login, altrimenti verrà effettuato il Listing richiamando il metodo:

```
Private Sub GetList(ByVal strUrl As String)
    Dim u As New Uri(strUrl)
    Dim listing As FtpList = FtpClient.List(u,
                                    Me.ftpUser, Me.ftpPwd)
    listing.DefaultSort()
    Me.result.InnerHtml = ""
    Dim sb As New System.Text.StringBuilder
    ...
    sb.Append("<table border=0 width=""100%""
                                    cellspacing='0'>")
    ...
    If Not listing.IsRoot Then
        FormatDirectory(sb, listing.ParentItem,
                                                True)
    End If
    For Each item As FtpListItem In listing
        FormatItemRow(sb, item)
    Next
    sb.Append("</table>")
    Me.result.InnerHtml = sb.ToString
End Sub
```

GetList richiama una serie di metodi definiti nello script della pagina volti a formattare l'output, da notare anche che per richiamare la funzione List di FtpClient abbiamo utilizzato una versione statica del metodo che avevamo approntato nella classe FtpClient. I metodi statici (shared in VB.NET) sono molto comodi perché permettono di richiamare al volo una funzione senza dover tutte le volte instanziare un nuovo oggetto.
Prima di effettuare il browsing della directory corrispondente all'URL corrente, OnLoad dovrà però valutare se tra le operazioni richieste c'è una cancellazione o un upload di file, queste operazioni vengono racchiuse in due metodi:

```
Private Sub EvalDelete()
    Dim delRequest As String = Request("del")
    If Not String.IsNullOrEmpty(delRequest) Then
        Try
            Dim u As New Uri(delRequest)
            Dim msg As String =
        FtpClient.DeleteFile(u, Me.ftpUser, Me.ftpPwd)
            msg = String.Format("Cancellazione file
                        {0} : {1}", u.ToString, msg)
            WriteClientMessage(msg)
        Catch ex As Exception
            WriteClientMessage(ex.Message)
        End Try
    End If
End Sub
    Private Sub EvalUpload()
    If Request.Files.Count > 0 Then
        Dim filename As String =
    IO.Path.GetFileName(Request.Files(0).FileName)
        If String.IsNullOrEmpty(filename) Then
            Return
        End If
        Dim reader As New
     IO.BinaryReader(Request.Files(0).InputStream)
        Dim b() As Byte =
  reader.ReadBytes(Request.Files(0).ContentLength)
        Try
            Dim u As New Uri(ftpUrl & "/" &
                                        filename)
            Dim msg As String =
    FtpClient.UploadFile(u, Me.ftpUser, Me.ftpPwd, b)
            msg = String.Format("Upload file {0} :
                            {1}", filename, msg)
            WriteClientMessage(msg)
        Catch ex As Exception
            WriteClientMessage(ex.Message)
        End Try
    End If
End Sub
```

EvalDelete funziona semplicemente esaminando se nella Request esiste il valore "del" che sarà corrispondente alla url del file da cancellare, se esiste richiama la funzione statica DeleteFile di FtpClient .
EvalUpload, invece, valuta se il flusso Request contenga o meno un Post di file ed eventualmente richiama la funzione statica UploadFile di FtpClient. Resta da vedere come viene gestita l'operazione di download del nuovo file.
Per effettuare il download è stato necessario approntare una nuova pagina ASP.NET,

DownloadFile.aspx, alla quale tutti i nomi dei file che compaiono nella tabella verranno linkati.
Il listato di DownloadFile.aspx è comunque molto semplice:

```
<%@ Page Language="VB" %>
<script runat="server">
  Public ReadOnly Property Url() As String
    Get
      Return Request("get")
    End Get
  End Property

  Protected Sub Page_Load(ByVal sender As Object,
                          ByVal e As System.EventArgs)
    Dim fStream As IO.Stream
    Try
      Dim u As New Uri(Url)
      fStream = FtpClient.DownloadFile(u,
                  Session("user"), Session("pwd"))
      Me.Response.ContentType = "application/x-
                                          download"
      Me.Response.AddHeader("content-
          disposition", "attachment; filename=" &
                          IO.Path.GetFileName(Url))
      Dim reader As New
                          IO.BinaryReader(fStream)
      Dim b(256) As Byte
      Dim count As Integer = reader.Read
                                      (b, 0, 256)
      While count > 0
        Me.Response.BinaryWrite(b)
        count = reader.Read(b, 0, 256)
      End While
      reader.Close()
      Me.Response.OutputStream.Close()

    Catch ex As Exception
      Me.Response.Clear()
      Me.Response.ContentType = "text/html"
      Me.Response.Write(ex.ToString)
      Me.Response.Write("<br>")
      Me.Response.Write("URL:" & Me.Url)
    End Try
  End Sub
</script>
```

In pratica in OnLoad si richiama la funzione statica DownloadFile di FtpClient e si legge il flusso che restituisce scrivendolo nella Response.
Da notare che il browser capisce che si tratta di un file da scaricare (e non di una pagina da mostrare) in quanto gli abbiamo passato, nella Response, un apposito Header content-disposition e un ContentType application/x-download.

LE ICONE

Resta da vedere come abbiamo ottenuto le icone delle directory e dei vari tipi di file senza dover approntare decine di GIFS.
Per fare questo in App_Code abbiamo predisposto un altro file, shFileIcon.vb, che utilizza l'API Win32 richiamando la funzione SHGetFileInfo di shell32.dll del sistema operativo, questa ci consente di ricavare la bitmap con l'icona del file passandogli semplicemente un nome di file con la relativa estensione (da notare che il file non deve essere fisicamente presente su disco, basta il nome). La stessa API consente anche di ottenere le bitmap delle icone utilizzate dal sistema operativo per le directory.
Abbiamo poi predisposto un'altra pagina ASP.NET chiamata icon.aspx il cui lavoro consiste unicamente di recuperare la bitmap dal nome di file passato nella Request e salvarla nel flusso Response:

```
...
    Dim bmpBase As New Bitmap(16, 16)
    Dim bmp As Bitmap
    If IsDirectory Then
      bmp =
iconMaker.GetFolderIcon(shFileIcon.IcoSizeEnum.smal
                                l, False).ToBitmap()
    Else
      bmp = iconMaker.GetFileIcon(Extension,
        shFileIcon.IcoSizeEnum.small).ToBitmap()
    End If
    Dim g As Graphics =
                    Graphics.FromImage(bmpBase)
    g.Clear(Color.White)
    g.DrawImage(bmp, 0, 0)
    Response.ContentType = "image/gif"
    bmpBase.Save(Response.OutputStream,
                    Imaging.ImageFormat.Gif)
...
```

Per ottenere la gif corrispondente nella pagina è sufficiente utilizzare icon.aspx come src di un tag HTML img :

```
<img src="icon.aspx?n=file.txt">
```

CONCLUSIONI

In questo articolo abbiamo visto come gestire il protocollo FTP in .NET 2.0.
Il nostro obbiettivo era un'applicazione web in grado di offrire un servizio di Client FTP, tuttavia le tecniche trattate e i listati allegati al Cd si prestano ad essere adattati e riutilizzati a tutte le situazioni in cui sia necessario utilizzare questo diffusissimo protocollo.

Francesco Smelzo

Francesco Smelzo è specializzato nello sviluppo in ambiente Windows con particolare riferimento ad applicazioni in ambiente .NET sia web-oriented che desktop. Il suo sito web è www.smelzo.it. Come sempre è a disposizione per ricevere suggerimenti o richieste sull'articolo all'indirizzo di posta elettronica francesco@smelzo.it

SINCRONIZZARE I DATI TRA SERVER E PALMARE

SE ABBIAMO UNA RETE DI PALMARI, I DATI PRESENTI IN UN DISPOSITVO DEVONO SICURAMENTE ESSERE SINCRONIZZATI CON UN DATABASE CENTRALE. SCOPRIAMO COME GESTIRE UPLOAD MULTIPLI EVITANDO EVENTUALI CONFLITTI

Nei precedenti due articoli, abbiamo affrontato buona parte delle problematiche che riguardano la realizzazione di software per dispositivi mobili. Si è parlato della scelta del device, della piattaforma, delle funzionalità e di come realizzare una interfaccia utente comoda da usare anche su schermi di piccole dimensioni.
Abbiamo anche visto come poter facilmente integrare il servizio web offerto da MapPoint per generare una mappa relativa all'indirizzo del cliente selezionato. Abbiamo però lasciato per ultima una esigenza molto sentita: la sincronizzazione dei dati.
Una applicazione come quella descritta negli articoli precedenti infatti, difficilmente sarà confinata al solo uso su dispositivi mobili. Molto più probabilmente, i dati usati dalla nostra applicazione, saranno generati da un sito web o da una applicazione gestionale aziendale e, i dati di ritorno dal palmare, dovranno essere riversati in un DataBase aziendale per poi essere elaborati.
Tale problematica ha alcuni lati oscuri e molto delicati, ma altrettante soluzioni utili alla corretta gestione della sincronizzazione.
In questo articolo vedremo quali sono le tecniche più comuni focalizzando l'attenzione su quella più economica e semplice da realizzare: Remote Data Access (o RDA).

Tempo di realizzazione

PERCHÈ ESISTE IL PROBLEMA DELLA SINCRONIZZAZIONE?

Se ci troviamo ad affrontare problematiche di sincronizzazione dei dati, siamo sicuramente di fronte ad un sistema composto da un DataBase centrale, localizzato su un server aziendale, ed uno o più DataBase remoti che non sono sempre connessi con quello centrale. È il caso dell'applicazione che accompagna questa serie di articoli. Il DataBase centrale conterrà i dati di tutti i clienti, di tutti i prodotti, gli ordini, le consegne, i fornitori etc., mentre quelli remoti, installati sui palmari degli agenti, conterranno solo una parte dei dati: quelli che servono all'agente per il suo lavoro. I dati però, ogni tanto, dovranno essere sincronizzati: quelli nuovi presenti sul DataBase centrale, come nuovi prodotti o aggiornamenti di prezzi, dovranno essere scaricati sui DataBase locali e gli ordini o eventuali modifiche alle anagrafiche, dovranno essere inviati dal client al server centrale.
E subito si possono individuare due importanti problematiche: l'accesso al DataBase centrale da parte del dipositivo ed i conflitti sui dati.
Il primo problema, apparentemente semplice, nasconde con se alcune insidie. Innanzitutto, come è buona norma, il DataBase centrale non sarà esposto su internet. Non è infatti detto che i client sincronizzino i dati dall'interno della lan aziendale. Bisogna quindi considerare tutti gli aspetti relativi alla sicurezza.
Il secondo problema invece è molto delicato e riguarda proprio l'attendibilità dei dati. Quello che può accadere infatti è che due o più sistemi remoti, quando sono disconnessi, modifichino lo stesso dato. Al momento della sincronizzazione, quale dato va accettato?
Facciamo un esempio pratico prendendo come esempio l'applicazione usata in questi articoli. Supponiamo che due rappresentanti, la mattina, sincronizzino i loro palmari scaricando, tra l'altro, l'anagrafica clienti, ed escano dall'ufficio per il loro consueto giro. Il rappresentante che chiameremo RA si reca dal suo cliente CA per prendere un ordine e, mentre è dal cliente, quest'ultimo gli comunica che il numero di telefono è errato. RA aggiorna l'anagrafica di CA sul suo palmare, fa l'ordine e saluta il cliente.
Per un errore di organizzazione, il rappresentante RB si reca da CA il quale, dopo avergli confermato che è già passato il suo collega, si ricorda di aver dato a RA un numero di telefono

errato e comunica ad RB quello corretto. Il nostro rappresentante aggiorna l'anagrafica di CA sul suo palmare e procede il suo giro.
Cosa accade la sera, quando i due agenti rientrano in sede ed aggiornano i dati sul server centrale? I dati di CA saranno stati modificati su ambedue i palmari. Qual è il dato che si troverà sul server? Per la corretta gestione di questa particolare problematica, esistono diverse strade. La più sensata, sicura e, se vogliamo, economica da gestire è la parzializzazione dei dati. Consiste nel fare in modo che due client non possano modificare lo stesso dato, parzializzando appunto i dati che ogni client gestisce. Nel caso della nostra applicazione d'esempio, ogni agente dovrebbe avere sul proprio palmare solo i clienti di competenza. In questo modo, RB non potrà mai aggiornare i dati di CA semplicemente perchè il dato non è presente sul proprio dispositivo. Un altra strada percorribile è quella di usare, per la sincronizzazione dei dati, il sistema di repliche di SQL Server. Tale sistema è estremamente potente e flessibile e consente di attuare delle dettagliate politiche di gestione dei conflitti. La sua trattazione però esula dallo scopo di questo articolo essendo un argomento abbastanza vasto e complesso da affrontare.
Una terza soluzione è quella di gestirsi autonomamente la gestione dei conflitti sui dati. Usando opportuni campi delle tabelle, possiamo sapere quando un record è stato modificato, se ci troviamo di fronte ad un conflitto e possiamo comportarci di conseguenza.
È evidente che tale sistema presuppone un grosso lavoro di progettazione e di implementazione. Ogni soluzione va ponderata in base alle reali esigenze e va attentamente analizzata. Nel prossimo paragrafo ne analizzeremo uno in particolare: RDA (Remote Data Access).

RDA: CONCETTI BASE

Remote Data Access è un sistema che consente di accedere e scambiare dati tra un dispositivo mobile su cui è installato SQL Server 2005 Mobile Edition ed un server centrale con SQL Server 2005. La grossa comodità di RDA è che, per la connessione al server, usa IIS risolvendo già implicitamente tutte le problematiche di raggiungibilità del server stesso.
In figura 1 possiamo vedere un tipico schema logico di come è possibile strutturare la rete per RDA.
In figura possiamo notare come i PDA si connettano alla rete lan aziendale sfuttando internet. Ciò rende possibile la sincronizzazione dei

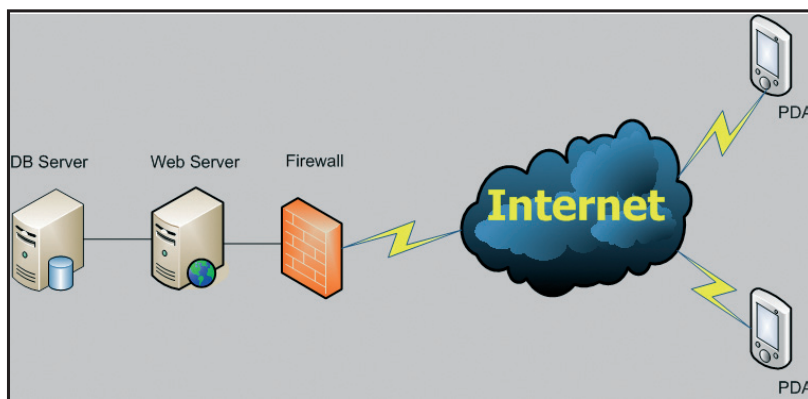



Fig. 1: **Architettura di una rete**

dati usando un qualsiasi mezzo di connessione, sia esso GPRS, UMTS, Wi-Fi, purchè l'indirizzo pubblico del servizio RDA sia raggiungibile. Sempre dalla figura è evidente come solo il server web può essere esposto su internet attraverso il firewall, lasciando il server contente il Data Base in una sotto rete separata e non esposta.
Internamente, RDA è strutturato in due moduli diversi: uno sul client, per gestire la comunicazione con il Data Base locale, ed un modulo server il cui scopo è quello di gestire le richieste del client.
In figura 2 è visibile uno schema sintetico di come è strutturato RDA.

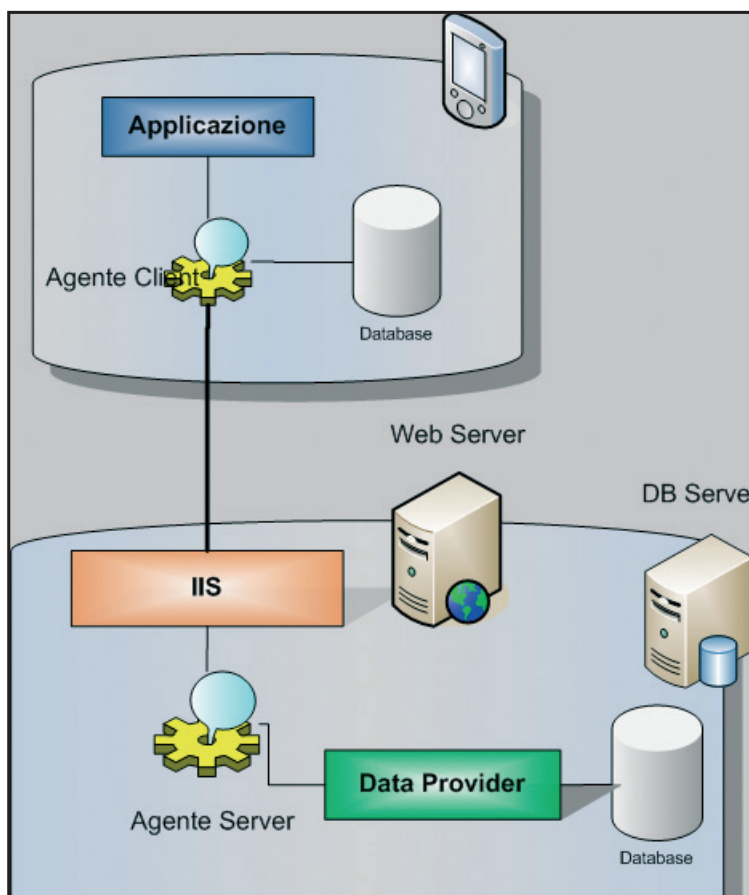



Fig. 2: **La struttura interna di RDA**

Lato client, l'agente locale si occupa di tracciare le modifiche ai dati delle tabelle che partecipano alla sincronizzazione. Quando uno dei metodi di RDA viene richiamato, l'agente locale contatta, attraverso IIS del server web, l'agente server che si occupa, attrraverso il data provider, di comunicare con il DataBase server.
RDA supporta sostanzialmente tre tipi di comandi molto semplici:

- Pull: le tabelle ed i dati da sincronizzare vengono scaricati dal server. I dati da scaricare sul client possono essere filtrati attraverso la clausola where della query da inviare al server. Possiamo così parzializzare i dati e fare in modo che ogni client ottenga solo quelli di sua competenza.
- Push: i dati modificati sul client vengono inviati al server ed inseriti nella tabella corretta.
- SubmitSql: da la possibilità di eseguire query SQL direttamente sul Data Base server, usando ovviamente l'architettura RDA.

È molto importante sottolineare una cosa: non è possibile richiamare il comando Push su una tabella di cui non è stato fatto prima il Pull, e questo perchè, senza aver eseguito il Pull, l'agente client non potrà tracciare le modifiche eseguite su una tabella. L'ordine di esecuzione da seguire quando si usa RDA è mostrato in figura 3.

Introduzione a RDA:
http://msdn2.microsoft.com/it-it/library/ms172029.aspx
Confronto tra RDA e Repliche:
http://msdn2.microsoft.com/it-it/library/ms172916.aspx
Configurazione e protezione:
http://msdn2.microsoft.com/it-it/library/ms172463.aspx
Utilizzo di RDA:
http://msdn2.microsoft.com/it-it/library/ms172917.aspx

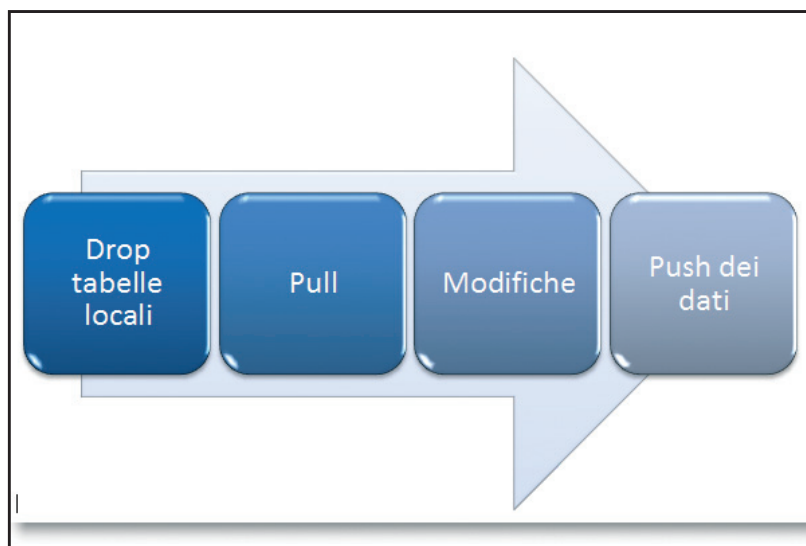
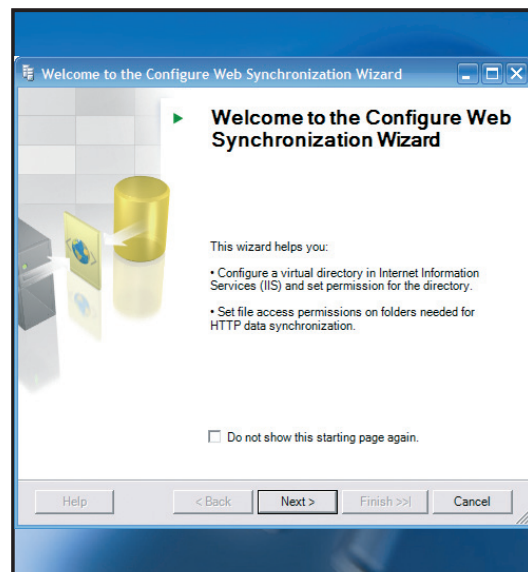
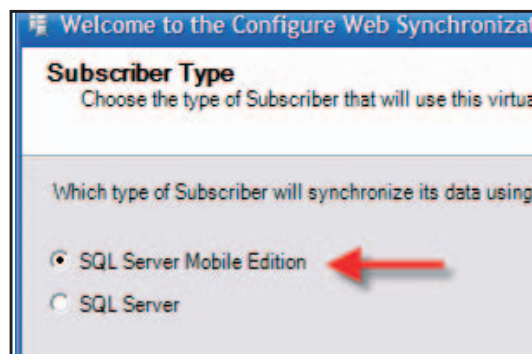
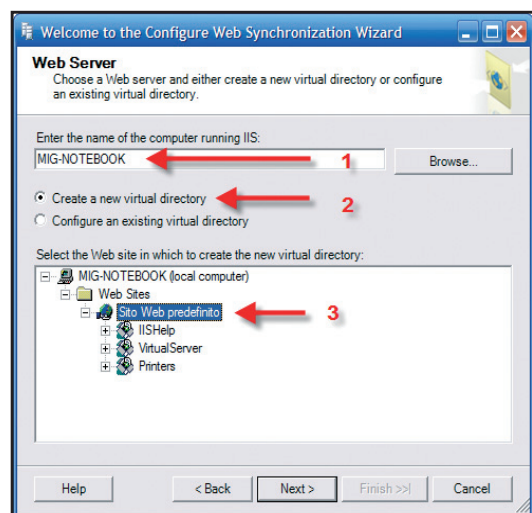
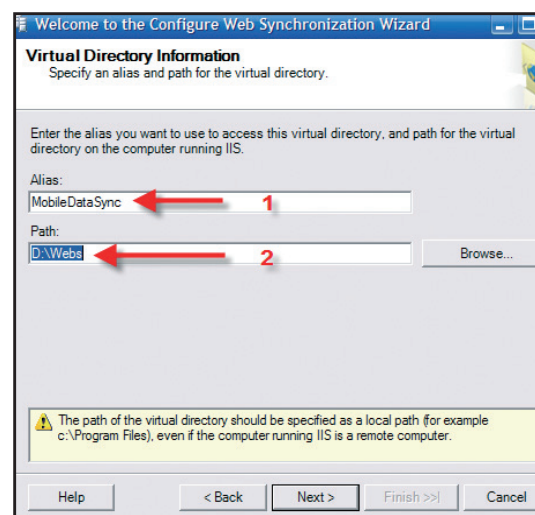
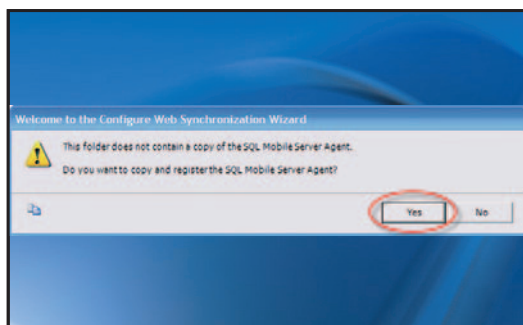
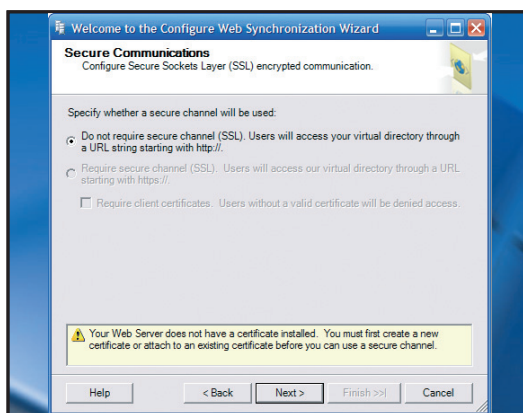



***Fig. 3:** L'ordine di esecuzione dei comandi con RDA*

Se l'ordine non viene rispettato, avremo degli errori nel momento in cui cercheremo di sincronizzare i dati con il server. Se abbiamo delle tabelle locali non presenti sul server, per cui non abbiamo la possibilità di eseguirne il Pull, ci torna comodo il comando SubmitSql.

INSTALLAZIONE E CONFIGURAZIONE

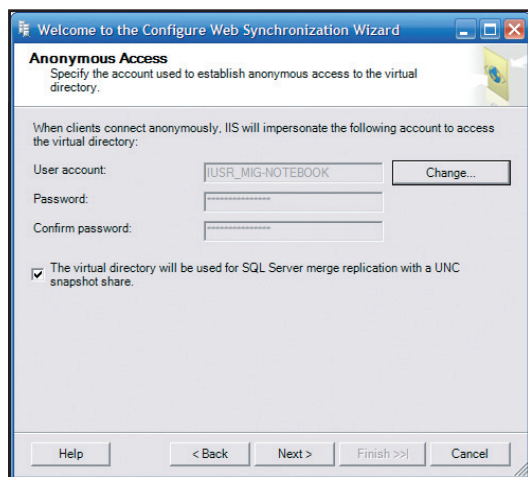
Prima di iniziare ad usare RDA però, dobbiamo installarne la parte server e configurala. La procedura non è immediata e richiede una serie di passi che vedremo in questo paragrafo.
Il primo passo è ovviamente quello dell'installazione della parte server. Dopo aver installato Microsoft SQL Server 2005, lanciamo il file sqlce30setupen.msi localizzato nella cartella C:\Programmi\Microsoft SQL Server\90\Tools\Binn\VSShell\Common7\IDE. Il Setup è abbastanza rapido e si completa facilmente in cinque passi. Una volta completato, bisognerà configurare i server tools. Di seguito la procedura completa.

1 Fare doppio click sul file ConnWiz30.exe presente nella cartella *C:\Programmi\Microsoft\SQL Server\90\Tools\Binn\VSShell\Common7\IDE.* Verrà avviata la procedura guidata di configurazione di RDA.

2 Spuntare la check box "SQL Mobile Edition" alla richiesta di che tipo di sottoscrizione vogliamo configurare.

3 Il terzo step ci permette di configurare correttamente il server web su cui è installato IIS. Tale server deve essere raggiungibile dall'esterno (vedi Figura 1). Per le nostre prove locali possiamo scegliere il PC di sviluppo su cui stiamo lavorando e specificare alla procedura di creare per noi il sito web. In caso stessimo configurando una applicazione reale, possiamo scegliere il nome della macchina su cui è installato IIS ed eventualmente un sito web già presente.

4 Se scegliamo la prima via, ci verrà chiesto di specificare un alias per il sito web (o directory virtuale) e la cartella sul file system in cui risiederanno i files

5 Se stiamo configurando RDA per la prima volta in un sito web, ci verrà segnalato che il componente che deve rispondere alle richieste provenienti da IIS (Sqlcesa30.dll) non è presente nella directory virtuale. Possiamo farlo copiare direttamente dal wizard cliccando su Ok.

6 per rendere sicura la comunicazione tra il client ed il server, abbiamo la possibilità di crittografare i dati scambiati lavorando sul protocollo https. Se abbiamo il certificato, possiamo scegliere questa opzione, altrimenti dobbiamo usare il normale protocollo http.

7 In questo step possiamo scegliere se il client ha bisogno di autenticarsi per eseguire la sincronizzazione dei dati. L'autenticazione può avvenire in svariati modi, ma la trattazione esula dallo scopo di questo articolo. Inutile dire che, in un ambiente reale, dovremmo scegliere https e l'autenticazione.

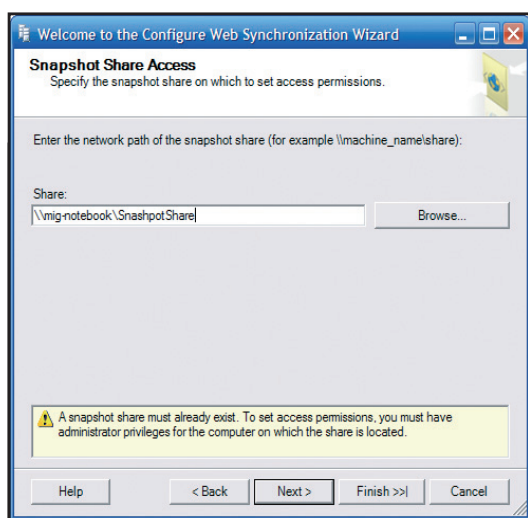
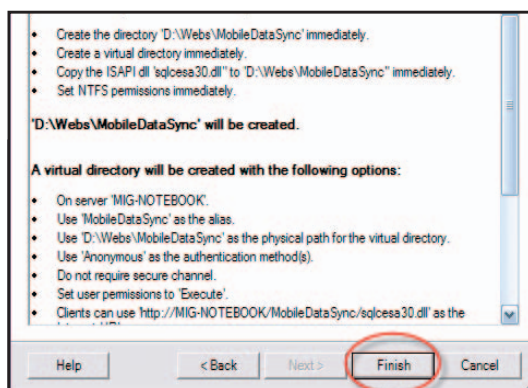
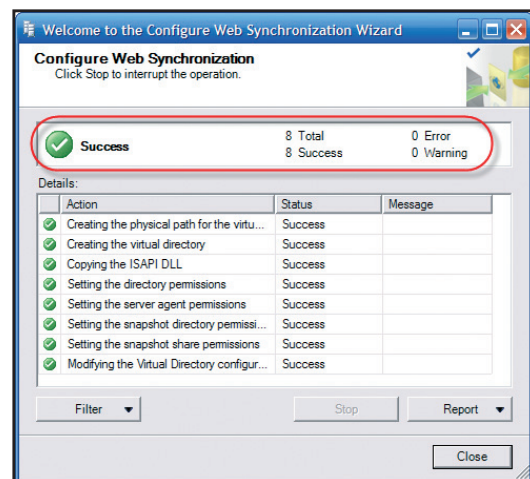
8 In caso di accesso anonimo, come nel nostro esempio, abbiamo la possibilità di scegliere con quali credenziali deve essere riconosciuto l'utente anonimo appunto. Lasciamo questa maschera così com'è e procediamo.

NOTA

DOVE TROVO IL WINDOWS MOBILE SDK?

Il Microsoft Windows Mobile 5.0 SDK è scaricabile al seguente indirizzo web: http://www.microsoft.com/downloads/details.aspx?familyid=83A52AF2-F524-4EC5-9155-717CBE5D25ED&displaylang=en ed include gli emulatori ed i template di Microsoft Visual Studio .NET 2005.

9 in questa maschera del wizard, dobbiamo specificare un path condiviso in cui appoggiare le tabelle di snashpot. Esse sono riferite alle repliche di Data Base, ma la procedura lo richiede e dobbiamo obbligatoriamente configurarle.

E SE HO SQL SERVER 2000?

Sebbene in questo articolo si parli di RDA con SQL Server 2005 e la relativa versione Mobile, anche in SQL Server 2000 e SQL Server CE è possibile usare, allo stesso modo, RDA.

10 L'ultima maschera che ci si presenta, riporta un report di quello che abbiamo impostato. Se è tutto ok, clicchiamo sul bottone "Finish" ed attendiamo che la procedura completi tutte le operazioni.

11 L'ultima maschera della procedura, ci conferma che tutte le operazioni sono state correttamente eseguite. Se qualcosa dovesse essere andato storto, ci verrà segnalato e sarà possibile usare l'help per correggere l'errore.

Una volta eseguiti tutti gli step elencati in questo paragrafo, possiamo immediatamente provare a vedere se tutto è stato correttamente configurato richiamando sul browser il seguente indirizzo: http://nome-pc/Mobile DataSync/sqlcesa30.dll (dove Nome-PC è il nome della vostra macchina di sviluppo o del server in cui è stato configurato il sito web e MobileDataSync è l'alias che abbiamo scelto al passo 4. Se il browser ci risponde con la dicitura: "SQL Server Mobile Server Agent 3.0" siamo pronti ad usare RDA. La stessa operazione è consiglabile farla usando il browser del pocket pc (o dell'emulatore).

DOWNLOAD DEI DATI: PULL

Come abbiamo visto in figura 3, se dobbiamo lavorare con RDA, la prima operazione da fare è la il Pull dei dati dal server verso il Data Base del dispositivo mobile. Essendo una operazione che potrebbe richiedere del tempo, per evitare di lasciare l'interfaccia bloccata, è meglio far eseguire la sincronizzazione in un thread separato e notificare, magari mediante dei messaggi in una textBox, lo stato di avanzamento della sincronizzazione.
Di seguito il codice per effettuare il Pull dei dati:

```
private void PullData() {
    //Pull delle Tabelle
    InvokeDelegate del = new
                        InvokeDelegate(InvokeMethod);
    SqlCeRemoteDataAccess rda = null;
    try {
        foreach (String table in _tablesName) {
```

```
            this.Invoke(del, new object[]{"Inizio
                                    drop tabella " + table});
            DBHelper.DropTable(table);
            this.Invoke(del, new object[]{"Drop
                              tabella " + table + " eseguito"});
            rda = new
                        SqlCeRemoteDataAccess(_ServerUrl,
                                   DBHelper.ConnString);
            this.Invoke(del, new object[] { "Inizio
                                    Pull tabella " + table });
            rda.Pull(table, "SELECT * FROM " +
                              table, _rdaOleDbConnectString,
RdaTrackOption.TrackingOn);
            this.Invoke(del, new object[] { "Pull
                           tabella " + table + " eseguito" });
        }
        this.Invoke(del, new object[] { "Procedura di
                                    Pull completata" });
    } catch (SqlCeException sqlCeEx) {
        MessageBox.Show("Si è verificato un errore
                 durante la sincronizzazione dei dati. Ripetere
                         l'operazione", "Mobile Order",
                                    MessageBoxButtons.OK,
                              MessageBoxIcon.Exclamation,
                         MessageBoxDefaultButton.Button1);
    } finally {
        rda.Dispose();
    }
}
```

_tableName è un vettore di stringhe contenente i nomi delle tabelle da sincronizzare. Ciclando gli elementi di questo vettore, richiamiamo come prima cosa un metodo del nostro DBHelper (si faccia riferimento al codice allegato per i dettagli) che esegue il Drop della tabella.
Viene successivamente creata una istanza dell'oggetto RD a cui si passa l'url del sito web che abbiamo configurato precedentemente e la stringa di connessione del Data Base locale. L'ultima operazione è quella di eseguire il Pull vero e proprio dei dati specificando i seguenti elementi:

1. Il nome della tabella di cui eseguire il Pull
2. La stringa sql per decidere quali dati della tabella scaricare sul client. In questa fase potremmo far scaricare sul device solo i dati di competenza di uno specifico agente.
3. La stringa di connessione al Data Base remoto (si veda il codice allegato)
4. Il tipo di tracking che vogliamo eseguire sulla tabella.

Il primo passo è completo. Ora il device può essere scollegato ed i dati possono essere modificati. Non ci resta che passare alla prossima operazione, ovvero l'invio dei dati sul server. Lo faremo con un'operazone di push dei dati

UPLOAD DEI DATI: PUSH

L'operazione opposta al Pull dei dati è il Push degli stessi verso il server centrale. Come per l'operazione di Pull, anche quella di Push può richiedere diverso tempo per essere eseguita. Si consiglia quindi di farla eseguire in un thread separato. Il codice per eseguire il push dei dati è il seguente:

```
private void PushData() {
    //Push delle tabelle
    InvokeDelegate del = new
                              InvokeDelegate(InvokeMethod);
    SqlCeRemoteDataAccess rda = null;
    try {
        foreach (String table in _tablesName) {
            this.Invoke(del, new object[] { "Inizio Push
                                    tabella " + table });
            rda = new
                        SqlCeRemoteDataAccess(_ServerUrl,
                                   DBHelper.ConnString);
            rda.Push(table, _rdaOleDbConnectString);
            this.Invoke(del, new object[] { "Push tabella
                           " + table + " eseguito" });
        }
        this.Invoke(del, new object[] { "Procedura di
                                    Push completata" });
    } catch (SqlCeException sqlCeex) {
        MessageBox.Show("Si è verificato un errore
                 durante la sincronizzazione dei dati. Ripetere
                         l'operazione", "Mobile Order",
                                    MessageBoxButtons.OK,
                              MessageBoxIcon.Exclamation,
                         MessageBoxDefaultButton.Button1);
    } finally {
        rda.Dispose();
    }
}
```

Anche in questo caso, l'operazione si riduce a poche righe di codice. Dopo aver creato una istanza di RDA, si richiama il metodo Push passando come argomenti il nome della tabella e la stringa di connessione del Data Base server. L'agente client di RDA si occuperà di inviare al server solo i record modificati dall'ultimo Pull della stessa tabella. Eventualmente, impostando opportuni flag, possiamo tenere anche traccia di eventuali errori di sincronizzazione. Si veda il box laterale per dei riferimenti sul web.

CONCLUSIONI

Termina qui la nostra serie sul Mobile. I tasselli per creare un'applicazione da zero ci sono tutti. Non vi resta che iniziare a sperimentare.

Michele Locuratolo

OTTIMIZZIAMO IL CODICE CON JRAT

I PROFILER SONO STRUMENTI MOLTO INTERESSANTII CHE CONSENTONO DI OTTENERE UNA MISURA DELLE PRESTAZIONI PRECISA MENTRE IL SOFTWARE È IN ESECUZIONE. SFRUTTANDO QUESTE TECNICHE È POSSIBILE OTTERE MIGLIORAMENTI SIGNIFICATIVI...

I profiler sono strumenti software in grado di analizzare il comportamento di un'applicazione a runtime ed estrarre utili informazioni quali il tempo di esecuzione dei metodi, l'occupazione di memoria, ecc... Scopo di tali misurazioni è quello di identificare i "colli di bottiglia" dell'applicazione per eseguirne refactoring mirati e ridurre così il tempo di esecuzione a tutto vantaggio delle prestazioni.
Il profiler che analizziamo in questo articolo è Jrat, pensato per Java ed open source. Jrat instrumenta le classi dell'applicazione in modo che ogni metodo al principio e al termine scriva su un log dei riferimenti temporali. L'interfaccia grafica di Jrat permette poi di visualizzare tale log sottoforma di un albero dove ogni nodo rappresenta una chiamata ad un metodo.
Per ogni metodo sono visualizzate varie informazioni tra le quali il tempo di esecuzione totale del metodo rispetto al tempo di esecuzione totale dell'applicazione, il tempo di esecuzione nell'ambito del metodo chiamante, il numero di volte che il metodo è terminato correttamente o ha sollevato un'eccezione.
Per utilizzare proficuamente JRat è necessario provarlo su un'applicazione vera e propria anche se di dimensioni limitate. Scegliamo come applicazione d'esempio XS/XXL Sudoku disponibile su internet all'indirizzo http://sudoku.danidemi.com. Questa è un'applicazione che permette di giocare a Sudoku su griglie da 4x4 sino a 25x25 passando per il classico 9x9.

PARTIRE CON JRAT

Il primo passo è quello di scaricare Jrat dalla relativa home page http://jrat.sourceforge.net/. Scaricate sia Jrat che BCEL, offerto nella stessa pagina di download. Jrat è il profiler vero e proprio. BCEL è una libreria Apache necessaria a Jrat per l'instrumentazione.

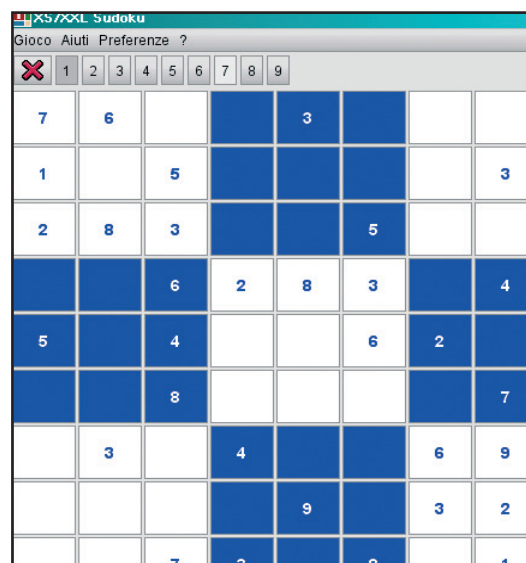



Fig. 1: **Per i nostri esempi tenteremo di ottimizzare l'applicazione del sudoku disponibile all'indirizzo http://sudoku.danidemi.com.**

Salvate il file di Jrat ed eseguite un doppio clic sul file appena scaricato. Quando l'applicazione vi chiederà la locazione del file BCEL navigate il file system e selezionate il file BCEL.jar scaricato insieme a Jrat. Il primo passo è quello di instrumentare le classi dell'applicativo per sottoporre ad analisi. Per fare ciò selezionare il seguente menù.

```
Instrument > Inject Directory Recursively
```

Scegliete ora la directory che contiene i file .class dell'applicazione, che potete trovare nella cartel*bin*. Lanciamo ora l'applicazione specificando i seguenti parametri.

```
java com.danidemi.sudoku.gui.Gui
        Djrat.factory=org.shiftone.jrat.provider.tree.
                              TreeMethodHandlerFactory
```

Utilizziamo normalmente l'applicazione, simulando una normale sessione, accedendo

a tutte le voci di menù e chiudiamo l'applicazione. Ora sempre da Jrat invochiamo la seguente voce di menù.

```
Jrat > Open Jrat Output File
```

Portarsi nella cartella JratOutput che è stata creata allo stesso livello della cartella “bin”che ha un nome simile al seguente:

```
JratOutput/2006-06-07_PM-11-44-02
```

Ovviamente il nome della cartella varierà in funzione del momento in cui avete fatto girare l'applicazione instrumentata. Identificate e selezionate il file con il seguente nome.

```
002_TreeMethodHandler.xrat.
```

Si aprirà quindi una finestra con il dettaglio del tempo di esecuzione dei vari metodi. I metodi “critici” che più occupano tempo sono indicati a mo di semaforo da una icona rossa.

UNA PRIMA OTTIMIZZAZIONE

Facendo qualche prova si può notare, sia attraverso i risultati di Jrat, sia con l'esperienza diretta, come la creazione di un nuovo schema di gioco sia la funzionalità più lenta. Allo scopo di avere un metro di paragone con il quale misurare la qualità delle ottimizzazioni che andremo ad apportare, riapriamo l'applicazione instrumentata, impostiamo la grandezza della griglia di gioco a 25x25 e per 5 volte richiamiamo la funzionalità di creazione di una nuova griglia, seguita ogni volta dalla risoluzione automatica della griglia stessa.
Apriamo il grafico Jrat. Uno dei metodi che impiega più tempo è **newGame()**. Scorriamo l'albero e notiamo che una prima ottimizzazione si potrebbe applicare al metodo **getDimension()**. Il sorgente di **getDimension()** è

```
public int getDimension() {
        return grid.length;
}
```

Grid è un array istanziato solamente nel costruttore della classe ConcreteGrid. Si potrebbe quindi precalcolare la lunghezza dell'array ed utilizzare direttamente tale valore risparmiando l'invocazione al metodo **getDimension()**.
Modifichiamo allora il costruttore in modo

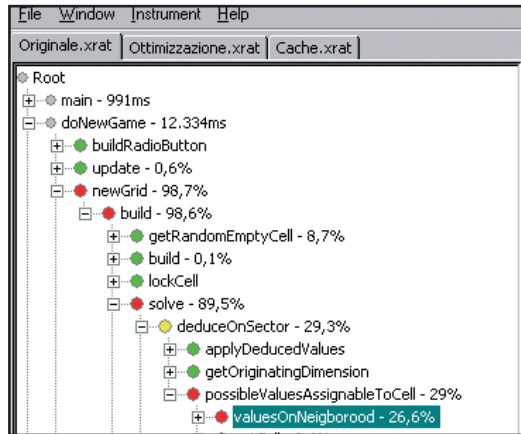



Fig. 2: **La schermata di Jrat mostra le prestazioni dell'applicativo. Il metodo doNewGame() è il più lento prendendo ben circa 12 secondi per creare 5 nuove griglie di gioco 25x25**

che memorizzi in un attributo di classe le dimensioni dell'array. Il costruttore originale, qui riportato:

```
grid = new CellValue
        [originatingDimension *
                originatingDimension]
        [originatingDimension *
                originatingDimension];
```

Sarà modificato in modo da memorizzare nel nuovo attributo di classe gridLength la dimensione dell'array

```
private final int gridLength;
...
grid = new CellValue
[originatingDimension *
                originatingDimension]
[originatingDimension *
                originatingDimension];
gridLength = grid.length;
...
```

Sostituiamo inoltre tutte le chiamate a **getDimension()** con un riferimento alla variabile gridLenght appena inserita.

OTTIMIZZAZIONE DI GETCELL()

Analizziamo il metodo **getCell()** di seguito riportato.

```
public Integer getCell(int x, int y) {
        checkAccessingCoordinate(x);
        checkAccessingCoordinate(y);
                if(grid[x][y]==null){
                throw new
```

```
IllegalArgumentException("cell " + x + "," + y + " is null.");
        }
        return grid[x][y].getValue();
    }
```

L'invocazione di **checkAccessingCoordinate()** pare inutile poiché il suo unico compito è quello di sollevare una IndexOutOfBoundException nel caso si cerchi di accedere ad una cella della griglia di gioco non esistente, e verrà quindi rimossa. Il doppio accesso all'array è inutile, se si memorizza il risultato del primo accesso per riutilizzarlo in seguito.Il codice si trasforma.

```
        CellValue cellValue = grid[x][y];
        if(cellValue==null){
            throw new
IllegalArgumentException("cell " + x + "," + y + " is null.");
        }
        return cellValue.getValue();
```

OTTIMIZAZIONE DI VALUESONSECTOR()

Proviamo ad ottimizzare il metodo valueOn **Sector()** e i metodi simili **valueOnRow()** e **valuesOnColumn()** che hanno il compito di restituire tutti i valori impostati rispettivamente sulla nel settore, nella riga e nella colonna insistenti sulla cella di coordinate x,

Fig. 3: Con le ottimizzazioni relativamente semplici descritte nei primi passi si ottiene una situazione in cui lo stesso metodo impiega solo all'incirca 3,5 secondi, migliorando la situazione iniziale di ben 4 volte.

y. I tre metodi eseguono un ciclo sulle celle della corrispondente zona raccogliendo i valori in un Vector().
In dettaglio il metodo **valuesOnSector()**, il più complesso.

```
    public Vector valuesOnSector(int x, int y)
{
        ...
        Vector numbers = new
Vector();
        int startX = (x /
            getOriginatingDimension())
                *
            getOriginatingDimension();
        int startY = (y /
            getOriginatingDimension())
                *
            getOriginatingDimension();
        for (int ix = startX; ix < startX
+
            getOriginatingDimension(); ix++) {
            for (int iy = startY; iy
<
        startY + getOriginatingDimension(); iy++) {
                if
                (getCell(ix,iy) != null) {
numbers.addElement(getCell(ix,iy));
                }
            }
        }
        return numbers;
    }
```

Il fulcro di questo metodo è il calcolo delle coordinate iniziali di un settore memorizzate nelle variabili startX e startY che viene ripetuto ad ogni invocazione. Si noti come tale calcolo ha lo scopo di ottenere partendo da una coordinata il valore di partenza del settore che contiene la coordinata stessa.
Prememorizziamo quindi le coordinate di inizio e fine di un settore all'interno di un array di interi. L'indice dell'array sarà la coordinata della griglia di gioco di cui vogliamo conoscere le coordinate di inizio del settore in cui è contenuta. Ad esempio in un griglia di gioco 9x9 la cella di indice 3 è compresa nel settore che parte dalla cella 2 e termina alla cella 4. Quindi l'array dovrà avere all'indice 3 valore 2. Un altro array potrebbe fornire la coordinata di fine del settore nel medesimo modo.
Popoliamo tali array nel costruttore, che saranno memorizzati negli attributi di classe coordTo StartSectorCoord[] e coordToEnd SectorCoord[].
Ne approfittiamo per inserire una variabile locale ed evitare la doppia chiamata a

getCell() all'interno del ciclo. Il sorgente si trasforma quindi in questa maniera.

```
private final int coordToStartSectorCoord[];
private final int coordToEndSectorCoord[];
//nel costruttore
coordToStartSectorCoord = new int[gridLength];
for (int i = 0; i < coordToStartSectorCoord.length;
                                                i++) {
        coordToStartSectorCoord[i] = (i /
    originatingDimension) * originatingDimension;
        ...
}

...
int startX = coordToStartSectorCoord[x];
int endX = coordToStartSectorCoord[y];
int endX = coordToEndSectorCoord[x];
int endY = coordToEndSectorCoord[y];

for (int ix = startX; ix < endX; ix++) {
        for (int iy = startY; iy < endY; iy++) {
                Integer value = getCell(ix, iy);
                ...
        }
}
```

SFRUTTIAMO LA CACHE

La classe principale dell'applicazione è ConcreteGrid che rappresenta la griglia del Sudoku e che realizza l'interfaccia Grid. Seguendo il design pattern decorator l'applicazione prevede altre implementazioni di Grid che offrono varie funzionalità.
Per accelerare l'esecuzione dei metodi scriviamo una nuova implementazione di Grid che abbia la responsabilità di eseguire il caching dei dati forniti dai metodi che restituiscono i valori presenti nelle righe, nelle celle e nei settori quali **valuesOnColumn()**, **valuesOnRow()** e **valuesOnSector()**. Ogni volta che uno di questi metodi viene invocato esso restituirà sempre lo stesso risultato, a patto che nessuna cella sia stata variata nella zona di competenza.
Ecco illustrato il funzionamento della classe. Ci sarà un riferimento alla Grid di cui si desidera effettuare il caching. Alla prima invocazione di un metodo di accesso alla griglia quale **valuesOnRow** l'esecuzione viene delegata alla griglia master. Il risultato viene memorizzato in una cache implementata tramite una Hashtable. Alla seconda e alle successive invocazioni il dato viene prelevato dalla cache senza delegarne il calcolo alla griglia master.
Ovviamente se dovesse essere aggiunto un valore in una cella facente parte della riga per cui si sono posti nella cache irisultati, la cache deve essere invalidata in modo che in un successivo accesso il calcolo venga nuovamente delegato alla griglia mater.

LA PRIMA BOZZA DELLA CLASSE

Viene implementata la classe Grid. Ci sono poi alcune costanti che identificano il dato in cache di una colonna, di un settore, o di una riga.

```
public class CachedGrid extends DelegationGrid
                                implements Grid {
        private static final int
                CACHE_NEIGHBOROUD = 3;
        public static int CACHE_ROW =  0;
        public static int CACHE_COLUMN =  1;
        public static int CACHE_SECTOR =  2;
        private Hashtable cache = new
                                Hashtable();
}
```

Creiamo una inner class che ci servirà come chiave della Hashtable. La chiave è composta da tre interi. Il primo identifica se si tratta della cache di una riga, di un settore o di una colonna. Il secondo intero identifica nei primi due casi l'indice della rica o della colonna. Nel terzo caso, insieme al terzo intero identifica le coordinate superiore sinistre del settore.

```
        private class CacheKey{
                private int cache = -1;
                private int c1= -1;
                private int c2 = -1;
                public CacheKey(int cache,
                        int c1, int c2) {
                        super();
                        this.cache = cache;
                        this.c1 = c1;
                        this.c2 = c2;
                }
                public boolean equals(
                        Object o){
                        ...
                }
                public int hashCode() {
                        ...
                }
                        }
```

In questo caso risulta importante implemen-

tare **equals()** e **hashCode()** per avere un utilizzo corretto della hashtable. In caso di invocazioni di un metodo che varia il valore di una cella, sia cancellandolo sia modificandolo, ovviamente il dato in cache per la riga, la colonna ed il settore che insistono sulla casella in questione vanno invalidati. I due metodi in questione si occupano di invalidare l'intera cache o solo quella dei dati insistenti su una cella.

```
private void invalidateCache(){
        cache.clear();
}
private void invalidateCache(int x, int y){
        cache.remove(new
        CacheKey(CACHE_COLUMN, x));
        cache.remove(new
                CacheKey(CACHE_ROW, y));
        cache.remove(new
CacheKey(CACHE_NEIGHBOROUD, x, y));
        cache.remove(new
                CacheKey(CACHE_SECTOR,
              master.getStartSectorCoord(x),
            master.getStartSectorCoord(y)));
}
```

I metodi che modificano le celle invalidano la cache tramite i metodi su esposti dopo aver delegato alla Grid decorata l'invocazione del metodo.

```
public void cancelCell(int x, int y) {
        super.cancelCell(x, y);
        invalidateCache(int x, int y)
}
public void clear() {
        super.clear();
        invalidateCache();
}
public void fillCell(int i, int j, int k) {
        super.fillCell(i, j, k);
invalidateCache(int x, int y)
}
```

Mentre i metodi che restituiscono i valori devono prima controllare la cache. Riportiamo l'esempio del più complesso valuesOnSector

```
public Vector valuesOnSector(int x, int y)
{
        int startX =
         master.getStartSectorCoord(x);
        int startY =
         master.getStartSectorCoord(y);
        CacheKey key = new
   CacheKey(CACHE_SECTOR, startX, startY);
        Vector result =
                (Vector)cache.get(key);
        if(result==null){
                result =
                  super.valuesOnSector(x, y);
                cache.put(key, result);
        }
        return result;
}
```

Lasciamo al lettore il compito di creare gli altri metodi.

Fig. 4: ***Inserendo la modifica architetturale del caching otteniamo una situazione in cui si passa da 12 secondi a circa 1 secondo ottenendo un risultato davvero utile dal punto di vista dell'utente.***

CONCLUSIONI

Rieseguendo ora l'applicativo con Jrat è possibile notare come le prestazioni siano notevolmente aumentate. Il codice è diventato più rapido ma si sono dovute inserire più variabili, aumentando quindi l'occupazione di memoria. Il codice ottimizzato ha la tendenza anche a diventare meno leggibile.
Tuttavia tramite l'uso di un profiler possiamo limitare questi interventi ai pochi punti che necessitano di ottimizzazione garantendo un buon equilibrio tra leggibilità e manutenibilità e le relative prestazioni.

Daniele De Michelis

JAVA COME PILOTA WORD COME MOTORE

SIAMO ABITUATI A PENSARE A JAVA COME UN MONDO COMPLETAMENTE STACCATO DA QUELLO DEI SOFTWARE CHE GIRANO SUI SISTEMI MICROSOFT. TUTTAVIA ESISTONO SOLUZIONI CHE POSSONO RENDERE I DUE MONDI MOLTO VICINI...

Siamo abituati a pensare a Java come un linguaggio i cui programmi possono essere eseguiti su qualsiasi piattaforma. Infatti il suo motto è: "write once, run everywhere" guadagnandoci la fama di (unico forse) linguaggio veramente indipendente dal sistema operativo. In questi ultimi anni però si sta affermando un nuovo modo di pensare alla programmazione nel quale diventa fondamentale che programmi scritti in linguaggi diversi riescano a comunicare fra di loro.
Questo aspetto non è sicuramente una novità visto che i Web Services (e non solo) sono nati proprio per favorire l'integrazione fra sistemi diversi. Uno dei primi e più pubblicizzati esempi di questa nuova filosofia si ritrova in .NET dove linguaggi diversi (C#, C++, Visual Basic) vengono, esattamente come accade in Java, compilati producendo un bytecode uguale per tutti i linguaggi.
Fra i vari intenti di .NET c'è anche quello di definire un bytecode standard a cui tutti i linguaggi (compreso forse Java, un giorno), debbano uniformarsi così da permettere a qualsiasi linguaggio di interagire con un programma .NET (a tal proposito è interessante dare un'occhiata a http://www.ikvm.net/).
Naturalmente questo è un discorso molto generico sull'interoperabilità fra i linguaggi; quello che invece spesso succede nella realtà è che si ha bisogno di interfacciarsi semplicemente con alcuni componenti del sistema operativo che possono essere degli eseguibili scritti in C o, su Windows, delle DLL o dei server COM.

DIFFERENZE FRA JNI AND JAWIN

Java in realtà offre già un modo per interfacciarsi ad eseguibili C e questa possibilità è offerta da JNI (ricordiamoci che Java stesso è scritto in C); in Java è già possibile utilizzare funzionalità offerte da eseguibili scritti in C e quindi anche componenti del sistema operativo, putroppo non sempre in modo semplice ed immediato.
In questo contesto trova spazio Jawin, un progetto opensource ospitato da Sourceforge.net, che si occupa di definire un'astrazione per l'interazione tra Java e alcuni componenti di Windows, quali oggetti COM e DLL Win32.

UN ESEMPIO CONCRETO

Entriamo nel merito di come utilizzare Jawin per interfacciarsi ad un oggetto scriptable qualunque. Le classiche operazioni che possono essere eseguite su un qualsiasi oggetto sono quelle di accedere ai suoi attributi (tramite metodi get e set) e quello di invocare dei metodi. Jawin ci offre una serie di metodi per questi scopi. In primo luogo bisogna ricordarsi di racchiudere il nostro codice tra le chiamate ai metodi:

```
Ole32.CoInitialize();
```

mentre in chiusura dovremo inserire

```
Ole32.CoUninitialize();
```

inoltre è buona norma ricordarsi di invocare il metodo **close()** su ogni puntatore ad oggetto COM che non viene più utilizzato; il primo passo da eseguire è ottenere un puntatore all'oggetto stesso identificato dal PROGID che nel caso di Office Word per esempio è: Word.Application. Il codice è:

```
DispatchPtr app = new
                DispatchPtr("Word.Application");.
```

Esposte queste premesse possiamo vedere qualè il metodo principale per il settaggio di

una proprietà di un oggetto:

```
public void put(String prop, Object val) throws
                                    COMException;
```

nel caso del riferimento alla Word Application definito prima e denominato app potremmo, aiutandoci anche con la documentazione msdn (http://msdn.microsoft.com/office /reference/vba/), settare la property Visible dell'oggetto stesso in modo da rendere visibile l'applicazione in questo modo

```
app.put("Visible", true);
```

potremmo leggere la stessa proprietà utilizzando il metodo:

```
public Object get(String prop) throws
                                    COMException;
```

ed in particolare con l'istruzione **app.get ("Visible")** che restituisce sempre un oggetto di tipo Object a cui bisogna applicare un cast in funzione del tipo di dato che ci si aspetta. Per quanto riguarda la conversione dei tipi di dati da COM a Java si può fare riferimento alla documentazione (javadoc) del metodo **Variant.ReadObject();** per quanto riguarda invece la chiamata a metodi il codice più generico e più usato è il seguente:

```
public Object invokeN(String meth, Object[] args)
                              throws COMException;
```

che accetta come parametri il nome del metodo ed un array di oggetti come parametri del metodo stesso. Come abbiamo visto ogni riferimento ad oggetto COM viene mappato da Jawin come DispatchPtr.
Applichiamo le conoscenze fin qui acquisite ad un esempio concreto. Se indaghiamo nella documentazione msdn ci accorgiamo che ActiveDocument è una property dell'oggetto Application ma per poter funzionare prima deve essere creato un nuovo documento; la creazione avviene invocando il metodo Add sulla property Documents dell'oggetto Application:

```
DispatchPtr app = new
                    DispatchPtr("Word.Application");
DispatchPtr docs =
                (DispatchPtr)app.get("Documents");
DispatchPtr doc =
                  (DispatchPtr)docs.invoke("Add");
```

Per creare una tabella 4X10 non dobbiamo fare altro che ottenere un'area selezionata dal documento e convertirla in tabella invocando il metodo **ConvertToTable()** dell'oggetto Selection che restituisce un oggetto Table.

```
doc.invoke("Select");
DispatchPtr win =
            (DispatchPtr)doc.get("ActiveWindow");
DispatchPtr sel =
                (DispatchPtr)win.get("Selection");
DispatchPtr table =
      (DispatchPtr)sel.invokeN("ConvertToTable",new
      Object[]{"",new Integer(10),new Integer(4)});
```

L'oggetto table quindi possiede un metodo Cell che accetta come parametri gli indici delle celle, quindi basta usare due cicli for annidati per scorrere le celle dell'intera tabella.

```
for (int j = 1; j <= 4; j++) {
for (int i = 1; i <= 10; i++) {
DispatchPtr cell =
            (DispatchPtr)table.invoke("Cell",new
                    Integer(i),new Integer(j));
-----------------------------
}
}
```

Ottenuto il riferimento alla cella da riempire e supponendo che il riferimento sia cell bisogna invocare il seguente metodo:

```
cell.Select();
cell.Selection.Range.InlineShapes.AddPicture()
```

MICROSOFT WINDOWS COMPONENT

COM è un framework, basato su RPC, definito da Microsoft nel 1993 con lo scopo di permettere la comunicazione tra processi. Un server COM non è altro che un componente software ospitato da una certa libreria in Windows (Dll) con cui si può comunicare conoscendo alcuni suoi identificatori che si trovano nel registro di Windows. I GUID (Global Unique Identifier) sono dei codici univoci a 128 bit utilizzati per identificare un componente COM; in particolare esso è identificato da un GUID chiamato CLSID, ma essendo tale codice alquanto difficile da ricordare sono stati definiti anche i PROGID, della forma <Program>.<Component>[.<Version>]. Esistono anche gli IID che identificano le interfacce implementate da ogni componente. Fra le varie interfacce che un componente COM può implementare esiste anche IDispatch che fornisce dei metodi quali GetTypeInfoCount, GetTypeInfo, GetIDsOfNames e Invoke che permettono di ottenere i dettagli dei metodi che il componente stesso espone e di richiamarli.
Passando alle dll esse sono una implementazione della libreria condivisa da parte di Microsoft; tali librerie espongono delle funzioni che possono essere invocate dai vari programmi lasciando al sistema operativo, fra le altre cose, il compito per esempio della gestione in memoria di questi componenti.

e passargli la url dell'immagine da caricare e questo lo facciamo con il seguente codice:

```
cell.invoke("Select");
sel = (DispatchPtr)win.get("Selection");
DispatchPtr rangeCell =
                    (DispatchPtr)sel.get("Range");
DispatchPtr shapes =
        (DispatchPtr)rangeCell.get("InlineShapes");
createBarcodeImage("123456789a","C:\\ws\\
                                jawin\\file.jpg");
DispatchPtr picture =
                    (DispatchPtr)shapes.invokeN("
                        AddPicture",new  Object[]{
            new String("C:\\ws\\jawin\\file.jpg")
                                              });
```

Come si può notare è stata inserita anche la chiamata ad un metodo per la generazione vera e propria dell'immagine, che come abbiamo accennato in precedenza si serve delle potenzialità della libreria barbecue; segue il codice del metodo:

```
private static void createBarcodeImage
                (String code,String jpgFilename){
// Always get a Barcode from the BarcodeFactory
Barcode barcode;
try {
barcode = BarcodeFactory.createCode128B(code);
barcode.setBarHeight(70);
barcode.setBarWidth(1);
FileOutputStream fos = new
                        FileOutputStream(jpgFilename);
BarcodeImageHandler.outputBarcodeAsJPEGImage
(barcode, fos);
catch (Exception e) {
e.printStackTrace();
}
}
```

PRO E CONTRO

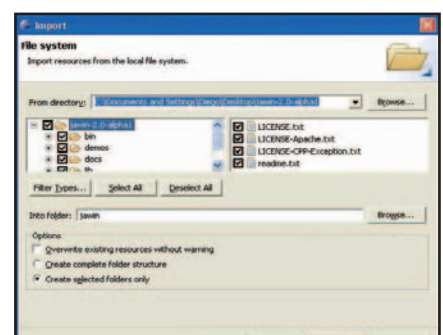
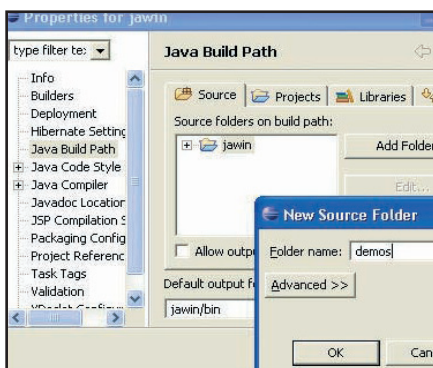
Probabilmente il codice che abbiamo presentato non è di stesura immediata ma la complessità sta comunque nel comprendere l'Object Model di Word: è questa l'unica difficoltà introdotta da questa tecnica che offre però infinite possibilità di interazione con i prodotti di Office. Naturalmente questo articolo non ha solo lo scopo di mostrare come stampare dei codici a barre ma ha la volontà è quella di porre in evidenza come si sia affermato il trend di trovare soluzioni per rendere gli innumerevoli linguaggi di programmazione sempre meno distanti gli uni dagli altri. E legaati al modello delle applicazioni già presente nel sistema. L'enorme diffusione del linguaggio della Sun e la natura open source di Jawin rendeno quest'ultimo un'ottima alternativa ad altre soluzioni pur sempre valide.

Diego Pansica

USARE JAWIN CON ECLIPSE

**Dopo aver scaricato dal sito http://sourceforge.net/projects/jawinproject il file zip jawin-2.0-alpha1.zip ed averlo decompresso è possibile creare un progetto su Eclipse che ci permetta di eseguire le demo con facilità.
Per fare ciò basterà seguire questi passi:**

- **creare un nuovo progetto Java su Eclipse creando una cartella denominata bin per l'output ed una cartella denominata demos per i sorgenti**
- **importare l'intera cartella di Jawin contenuta nella distribuzione di jawin**
- **importare le librerie che si trovano nella cartella lib e le librerie di barbecue**
- **copiare il file jawin.dll sotto C:\Windows\System32 oppure aggiungerlo al path di Windows**

Jawin offre la possibilità di interagire con oggetti COM scriptable e non; noi ci limiteremo ad interagire con un oggetto scriptable cioè con oggetto che implementa l'interfaccia IDispatch;

in particolare dimostreremo come utilizzare Word per stampare una serie di codici a barre in un determinato layout; Java ci permetterà di generare i codici a barre a partire da dei codici alfanumerici (utilizzando la libreria barbecue) mentre Word ci darà tutte le potenzialità di paginazione e ci eviterà di preoccuparci della stampa.

FACILITARE IL DEBUG CON I VISUALIZER

NON TUTTI SANNO CHE VISUAL STUDIO 2005 OFFRE POSSIBILITÀ DI PERSONALIZZAZIONE PRATICAMENTE INFINITE. IN QUESTO ARTICOLO ILLUSTREREMO COME MIGLIORARE IL DEBUG DI STRUTTURE PROPRIETARIE, UTILIZZANDO CONTROLLI SPECIFICI

Chi sviluppa sa benissimo che la fase di debug è fondamentale, necessaria per scovare e correggere tutti gli eventuali errori che abbiamo inserito nel codice. Le interfacce utente degli ambienti di sviluppo moderni ci vengono in aiuto, fornendoci una serie di strumenti grazie ai quali possiamo esaminare i valori delle variabili e lo stato degli oggetti in corrispondenza dei breakpoint che abbiamo impostato. Visual Studio 2005 non può essere da meno. Il menu Debug disponibile nell'IDE permette di attivare tutta una serie di finestre dedicate al debug, che ci permettono di controllare per esempio l'elenco dei breakpoint impostati, di vedere il contenuto delle variabili definite nello scope corrente, di consultare lo stack e i thread in esecuzione.
.NET ci dà però una marcia in più, consentendo a ciascuno di noi di sviluppare speciali visualizzatori associati ad un particolare tipo di oggetto .NET. Tali visualizzatori sono comunemente chiamati visualizer. È così possibile creare visualizer specifici per le stringhe, per le immagini, per gli array, per i nostri business object ed in generale per qualsiasi oggetto .NET ci interessi. Il visualizer è chiamato in causa esclusivamente durante il debug della nostra applicazione, ed in diversi contesti: quando spostiamo il puntatore del mouse sopra una variabile il cui tipo è quello previsto dal visualizer, oppure quando la stessa variabile appare in una delle finestre di Visual Studio 2005 dedicate al debug (Autos, Locals, Watches, etc.).
Lo scopo di un visualizer è quello di fornire un'interfaccia per esaminare l'oggetto, migliore rispetto a quanto prevede l'IDE, che per default ci fa esplorare l'intera gerarchia delle proprietà usando un controllo che ci ricorda la TreeView.
In questo articolo vedremo come creare un visualizer da zero, fornendo alcuni spunti interessanti per poterlo migliorare in seguito.

CREARE PASSO PASSO IL NOSTRO PRIMO VISUALIZER

Possiamo creare un visualizer per qualsiasi oggetto .NET, creato con qualsiasi linguaggio managed. In questo articolo, vedremo come creare un visualizer specifico per la classe ArrayList. Ogni volta che nel nostro codice comparirà un oggetto di questo tipo, potremo utilizzare il nostro visualizer per vederne il contenuto o, addirittura, per poterlo modificare in debug. Passiamo alla pratica. Iniziamo innanzitutto con la creazione di un nuovo progetto. Apriamo Visual Studio 2005, andiamo sul menu *File/New* e poi clicchiamo sulla voce Project. Selezioniamo nella parte destra il tipo di progetto Class Library, chiamiamolo *IoProgrammoCustomVisualizer* e confermiamo il tutto cliccando sul pulsante Ok. Rinominiamo la classe con il nome ArrayList Visualizer. A questo punto è necessario aggiungere un riferimento alla libreria **Microsoft. VisualStudio.Debugger Visualizers**. Possiamo farlo selezionando il nodo **References** raggiungibile all'interno del progetto **IoProgrammo CustomVisualizer**, cliccando con il pulsante destro e selezionando la voce **Add Reference**. Nella finestra di dialogo è sufficiente ricercare il file all'interno della scheda denominata .NET. Aggiungiamo un riferimento anche al file **System.Windows.Forms**, in modo da poter usare le Windows Forms nel nostro visualizer ed utilizzare classi come la **MessageBox.** Fatto questo, possiamo tornare al codice della nostra classe **Array ListVisualizer**. Assicuriamoci innanzitutto di referenziare attraverso l'istruzione using tutti i namespace che ci serviranno nel codice:

```
using System;
using System.Collections;
using System.Windows.Forms;
using Microsoft.VisualStudio.DebuggerVisualizers;
```

In secondo luogo, la dichiarazione della nostra classe deve comparire come segue.

```
namespace IoProgrammoCustomVisualizer
```

```
{
    public class ArrayListVisualizer :
                                    DialogDebuggerVisualizer
    { }
}
```

La classe **ArrayListVisualizer** eredita direttamente dalla classe **DialogDebuggerVisualizer**.
Quest'ultima classe mette a disposizione tutta l'infrastruttura .NET necessaria per la creazione del visualizer. Giunti a questo punto, buona parte del lavoro è fatto. Ci basta fare l'overloading del metodo **Show()** per poter avere un visualizer funzionante. Il secondo parametro del metodo **Show()** è il più interessante, perché è attraverso questo che riusciremo ad avere un riferimento all'oggetto di cui stiamo facendo il debug. Il codice sottostante mostra un semplice esempio di come implementare il metodo **Show()**:

```
protected override void Show
(IDialogVisualizerService windowService,
                    IVisualizerObjectProvider objectProvider)
{
    ArrayList lst =
                    (ArrayList)objectProvider.GetObject();
    MessageBox.Show(lst.Count.ToString());
}
```

Attraverso il metodo **GetObject()** esposto dalla classe **IVisualizerObjectProvider**, otteniamo una copia dell'oggetto che ci interessa debuggare, nel nostro caso l'**ArrayList**. Nel semplice esempio qui sopra, visualizziamo semplicemente il numero di elementi contenuti nell'**ArrayList** e nulla di più. Sebbene sia un caso estremamente semplice, rende l'idea e descrive la struttura minima per realizzare un visualizer funzionante. All'interno del metodo **Show()** possiamo scrivere tutto il codice che ci serve, come mostrare una Windows Form: ne parleremo più avanti in questo articolo. Adesso vediamo come testare il visualizer e controllare che tutto funzioni correttamente.

TESTIAMO IL VISUALIZER APPENA CREATO

Abbiamo detto prima che il visualizer appare quando, durante il debug, spostiamo il puntatore sopra l'oggetto interessato, oppure quando l'oggetto stesso compare nelle finestre di debug di Visual Studio 2005. Per poter attivare questo meccanismo, dobbiamo decorare la classe **ArrayListVisualizer** con un attributo, che vedremo successivamente. Per adesso, la cosa ottimale è testare il nostro visualizer attraverso un metodo statico **TestShowVisualizer**, da aggiungere alla classe **ArrayListVisualizer**, che verrà poi rimosso nella versione definitiva:

```
public static void TestShowVisualizer(object
                                        objectToVisualize)
{
    VisualizerDevelopmentHost vHost = new
VisualizerDevelopmentHost(objectToVisualize,
                            typeof(ArrayListVisualizer));
    vHost.ShowVisualizer();
}
```

Il parametro in ingresso al metodo rappresenta l'oggetto che vogliamo debuggare. Per testare il visualizer, non dobbiamo fare altro che chiamare questo metodo: tale operazione scatenerà l'esecuzione del metodo **Show()** descritto precedentemente.
Compiliamo quindi la classe **ArrayListVisualizer** per ottenere l'assembly pronto per l'uso. Aggiungiamo un nuovo progetto Windows Application e chiamiamolo TestVisualizer: questo sarà il nostro piccolo progetto di test. Visual Studio 2005 crea automaticamente una Windows Form completamente vuota, alla quale aggiungiamo un Button chiamato **btnCreaArrayList** e gestiamone l'evento Click: qui possiamo semplicemente creare un oggetto ArrayList ed aggiungere un po' di elementi. Ad esempio, prendiamo in considerazione il seguente codice:

```
private void btnCreaArrayList_Click(object
                                    sender, EventArgs e)
{
        ArrayList lst = new ArrayList();
                for (int i = 0; i < 100; i++)
                lst.Add(i);
        ArrayListVisualizer.TestShowVisualizer(lst);
        MessageBox.Show("Esecuzione terminata!");
}
```

Il codice qui sopra crea una nuova istanza di ArrayList, al quale aggiungiamo semplicemente i primi 100 numeri interi. Notare la linea [6], nella quale non facciamo altro che chiamare in causa il visualizer, usando il metodo **TestShowVisualizer**, passandogli come parametro l'oggetto lst, che non è

LA SERIALIZZAZIONE IN .NET

La serializzazione è una tecnica utilizzata frequentemente in .NET, e consiste nel persistere la struttura e lo stato di un oggetto managed in un formato standard XML. Tale conversione può essere controllata da una serie di attributi che, applicati a dovere sulle classi e sulle proprietà, ci permettono di decidere la struttura del documento XML ed i nomi dei tag. Su MSDN è pubblicato un documento aggiornato al Framework 2.0 che descrive nel dettaglio come usare questi attributi: http://msdn2.microsoft.com/en-us/library/2baksw0z.aspx.

nient'altro che l'ArrayList che vogliamo debuggare. Per controllare meglio il flusso del codice, possiamo anche impostare un breakpoint alla linea [6]. Dopo aver compilato tutta la soluzione, avviamo il progetto TestVisualizer, clicchiamo sul bottone ed aspettiamo che .NET faccia il resto. Se abbiamo fatto tutto correttamente, dovremmo veder comparire una MessageBox che mostra il numero di elementi nel nostro ArrayList, a dimostrazione del fatto che il visualizer è stato eseguito correttamente. Ecco il nostro primo custom visualizer! Ora non ci resta che migliorarlo e renderlo un po' più utile. Non dobbiamo fare altro che tornare al progetto class library IoProgrammoCustom Visualizer e modificare il metodo **Show()** e decidere cosa vogliamo fargli fare.

IL NAMESPACE MICROSOFT. VISUALSTUDIO.DEBUGGERVISUALIZERS

Scrivere un visualizer significa prima di ogni altra cosa lavorare con classi contenute nel namespace Microsoft.Visual Studio,DebuggerVisualizers, che contiene l'implementazione delle classi e delle interfacce necessarie, come Dialog DebuggerVisualizer che è la classe dalla quale il nostro visualizer deve ereditare. Potete avere maggiori informazioni partendo dall'indirizzo http://msdn2.microsoft.com/en-us/library/microsoft.visualstudio.debuggervisualizers.aspx.

UTILIZZARE UNA WINDOWS FORM NEL NOSTRO VISUALIZER

Una cosa che potrebbe tornare molto comoda è creare una normale Windows Form che possa in qualche modo mostrare ogni singolo elemento contenuto nell'ArrayList. Possiamo pensare ad esempio di utilizzare uno dei controlli .NET che meglio si adattano in contesti di questo tipo, come una semplicissima ListBox.

Non dobbiamo fare altro che aggiungere una Windows Form (che chiamiamo ArrayList Content Form) nel progetto IoProgrammo CustomVisualizer, posizionare all'interno di essa una ListBox, chiamandola lstArrayListContent, ed utilizzarla all'interno del metodo **Show().**

Il codice apparirà come di seguito:

```
protected override void Show
(IDialogVisualizerService windowService,
                IVisualizerObjectProvider objectProvider)
{
            ArrayList lst =
                    (ArrayList)objectProvider.GetObject();
            ArrayListContentForm frm = new
                                    ArrayListContentForm();
            frm.ArrayListToView = lst;
            windowService.ShowDialog(frm);
}
```

Il codice qui sopra non fa altro che inizializzare il form, ottenere un oggetto di tipo ArrayList e visualizzare la Windows Form attraverso il metodo **ShowDialog** esposto dalla classe **IDialog VisualizerService.** Quello che resta da fare è implementare qualche linea di codice anche sulla Windows Form. Innanzitutto ci serve una proprietà custom ArrayListToView, che viene utilizzata nel metodo **Show()** per il passaggio di parametri. In secondo luogo, gestiamo l'evento Load della nostra Windows Form per attivare il databinding tra l'ArrayList e la ListBox, passando attraverso un oggetto **BindingSource**, raccomandato ogniqualvolta si faccia databinding con .NET 2.0:

```
private void ArrayListContentForm_Load(object
                                    sender, EventArgs e)
{
    bsSource.DataSource = _arrayListToView;
    this.lstArrayListContent.DataSource = bsSource;
}
```

Giunti a questo punto, possiamo ricompilare la soluzione e provare ad eseguire nuovamente il progetto. Il visualizer questa volta fa una cosa molto diversa, e molto più utile: sullo schermo appare la Windows Form che abbiamo appena creato, che ci mostrerà il contenuto del nostro ArrayList. Questo non solo ci permette di dare un'occhiata all'ArrayList in fase di debug, ma ci apre la strada per creare nuove interessanti funzionalità, come il permettere la modifica dell'ArrayList stesso. Per fare questo, però, non dobbiamo più usare il metodo di test TestShowVisualizer, ma dobbiamo andare un po' più in là, sfruttando appieno l'integrazione offerta da Visual Studio 2005.

DEPLOY E COMPLETA INTEGRAZIONE DEL NOSTRO VISUALIZER

Il visualizer in realtà offre molto di più rispetto a quanto abbiamo visto finora. La vera potenza di un visualizer sta nella sua piena integrazione con l'ambiente di sviluppo, cosa che finora non abbiamo preso in considerazione. Per poterlo veramente sfruttare al 100%, dobbiamo decorare la classe ArrayListVisualizer con un attributo che fornisce ulteriori informazioni riguardanti il visualizer che abbiamo sviluppato. L'attributo in questione è DebuggerVisualizerAttribute, e viene utilizzato nel modo seguente:

```
[assembly:
      System.Diagnostics.DebuggerVisualizerAttribute(
```

```
typeof(IoProgrammoCustomVisualizer.ArrayList
    Visualizer), typeof(VisualizerObjectSource),
Target = typeof(ArrayList), Description = "Visualizer
    per ArrayList")]
```

L'attributo **DebuggerVisualizerAttribute** decora la classe **ArrayListVisualizer**, impostando per esempio una descrizione e comunicando a .NET il tipo al quale si associa il visualizer stesso: questo è lo scopo del parametro Target. Dopo aver ricompilato l'intera soluzione, possiamo tranquillamente distribuire il visualizer, che è contenuto nell'assembly ArrayListVisualizer.dll. È sufficiente copiare questo file nella directory *\Documenti\Visual Studio 2005\Visualizers* per poter utilizzare il visualizer in modo molto più veloce di quanto abbiamo fatto finora. Se avete copiato l'assembly nella directory indicata, chiudete e a riaprite l'IDE. All'avvio, Visual Studio 2005 carica tutti i visualizer contenuti in *\Documenti\Visual Studio 2005\ Visualizers*, compreso il nostro ArrayList Visualizer. Dal momento che non vogliamo più utilizzare il metodo **TestShowVisualizer**, possiamo tranquillamente cancellare la linea [6] che usava il metodo **TestShowVisualizer**. Il codice che ci serve è il seguente:

```
private void btnCreaArrayList_Click(object
    sender, EventArgs e)
{
    ArrayList lst = new ArrayList();
        for (int i = 0; i < 100; i++)
        lst.Add(i);
    MessageBox.Show("Esecuzione
        terminata!");
}
```

Impostiamo il breakpoint alla linea [6], che adesso contiene la MessageBox ed eseguiamo il progetto. Quando entriamo in modalità di debug e l'esecuzione del codice si interrompe, spostiamo il puntatore del mouse sopra l'oggetto lst: Visual Studio 2005 mostra un piccolo tooltip che mette in evidenza l'oggetto, permettendoci di esplorarlo come solito. Questa volta però nel tooltip vediamo anche una piccola icona: una lente di ingrandimento.
La presenza di questa icona indica che a questo tipo di oggetto è associato un visualizer. Se la clicchiamo, non facciamo altro che attivare il nostro ArrayListVisualizer, esattamente come accadeva prima, ma senza dover scrivere codice dedicato al testing e soprattutto in modo molto più integrato, anche in virtù del fatto che il visualizer è disponibile, come dicevamo prima, in tutte le finestre di Visual Studio 2005 dedicate al debug (Watches, Autos e così via.). Quello che abbiamo visto è la modalità migliore per usare un visualizer, ed è la stessa modalità con la quale il nostro visualizer verrà utilizzato dagli altri sviluppatori se decidessimo di distribuirlo. Se tutto quello che volevamo era creare un visualizer, potremo fermarci qui ma noi però vogliamo andare oltre, e permettere la modifica dell'oggetto che stiamo debuggando.

COMPLETARE IL VISUALIZER: MODIFICARE L'ARRAYLIST IN DEBUG

In questo momento, abbiamo costruito un visualizer funzionante in tutto e per tutto. Quando entriamo in modalità debug, il visualizer ci permette di esaminare il nostro oggetto, ma senza poterlo modificare. Supponiamo di voler fare un ulteriore passo in avanti, cioè di permettere allo sviluppatore di avere più controllo: magari vorrebbe svuotare completamente l'ArrayList. Come possiamo implementare questa nuova funzionalità? Non dobbiamo fare altro che modificare l'implementazione della Windows Form, per esempio aggiungendo un Button (chiamato btnClear AllItems) dedicato a questo scopo:

```
private void btnClearAllItems_Click(object sender,
    EventArgs e)
{
    _arrayListToView.Clear();
    bsSource.ResetBindings(false);
}
```

Quando lo sviluppatore cliccherà il Button, il contenuto dell'ArrayList verrà completamente eliminato. La seconda linea contenuta provoca il refresh della ListBox. E se invece volessimo metterlo in condizione di modificare uno ad uno ogni elemento dell'ArrayList?
Sarà sufficiente gestire l'evento **SelectedIndex Changed** della ListBox e visualizzare l'elemento selezionato in una TextBox, così da poterne modificare il valore:

```
private void
    lstArrayListContent_SelectedIndexChanged(object
        sender, EventArgs e)
{
    if (lstArrayListContent.SelectedIndex == -1)
        return;
    int number =
        (int)_arrayListToView[lstArrayListContent.S
        electedIndex];
    txtSelectedItem.Text = number.ToString();
}
```

All'evento **Validating** della TextBox, tentiamo di convertire in intero ciò che l'utente ha scritto e, se è possibile, aggiorniamo l'ArrayList con il nuovo valore. Manca ancora un piccolo particolare importante, che fino ad ora abbiamo omesso. All'interno del codice del visualizer gestiamo un'istanza di ArrayList che è una copia di quella contenuta nel codice debuggato: la chiamata al metodo **GetObject**() nel metodo **Show**() serve proprio a trasferire l'oggetto da un contesto verso l'altro. Quando lo sviluppatore chiude la Windows Form, dobbiamo fare l'operazione esattamente contraria, ovvero restituire al codice del progetto di test l'ArrayList eventualmente modificato. Questa operazione è facilmente ottenibile attraverso il metodo ReplaceObject esposto dalla classe IVisualizer ObjectProvider:

```
if (res == DialogResult.OK)
{
    objectProvider.ReplaceObject(lst);
}
```

In questo modo, se la Windows Form viene chiusa con il Button Ok, trasferiamo l'istanza dell'ArrayList che viveva all'interno del visualizer al codice chiamante, che nel nostro caso si tratta del piccolo progetto di test ma che in realtà sarà la nostra applicazione reale che stiamo sviluppando. Possiamo testare queste nuove funzionalità semplicemente rilanciando l'esecuzione del progetto di test: al momento dell'attivazione del visualizer, possiamo modificare l'ArrayList come meglio crediamo, utilizzando i nuovi controlli che abbiamo posizionati sulla Windows Form. Se chiudiamo la form con il Button Ok, l'ArrayList che abbiamo modificato nel visualizer sostituisce l'ArrayList del progetto di test, rappresentato dalla variabile lst.
Questa è la vera potenza di un visualizer, ovvero avere l'assoluta capacità di manipolare un oggetto in debug, esattamente come succede con variabili value-type, ma con tipi più complessi.

CREARE UN VISUALIZER PER I NOSTRI BUSINESS OBJECT

In questo articolo abbiamo descritto come creare un visualizer per una classe .NET standard come l'ArrayList. Come è facile immaginare, possiamo applicare le stesse logiche descritte in questo articolo con qualsiasi oggetto, compresi i business object appartenenti al domain model delle nostre applicazioni. Costruire un visualizer è tanto più utile quanto più i nostri oggetti sono riutilizzabili: se stiamo costruendo un software di fatturazione, ad esempio, potremmo pensare di realizzare visualizer per oggetti del domain model come clienti, fatture o prodotti: durante lo sviluppo del software, tali visualizer ci consentiranno una maggior intuitività nel debug, verificando velocemente che gli oggetti siano nello stato che ci si aspetta, e magari correggendo i valori di alcune proprietà. L'unico requisito richiesto per sviluppare un visualizer per un certo tipo di oggetto è che tale oggetto sia serializzabile. La serializzazione è una tecnica piuttosto comune in .NET, e consiste nel rappresentare la struttura e lo stato di un oggetto attraverso XML. Queste consente il trasferimento dell'oggetto stesso da un contesto all'altro, da un'applicazione all'altra, e così via. Molte classi .NET sono naturalmente serializzabili, come l'ArrayList di cui abbiamo parlato prima. Dobbiamo quindi assicurarci di poter serializzare i nostri business object, decorando la classe con l'attributo [Serializable], e così per la proprietà esposte dall'oggetto stesso. Potete trovare maggiori informazioni sulla serializzazione nel box a lato.

I DUE ASPETTI DI UN VISUALIZER: DEBUGGER E DEBUGGEE

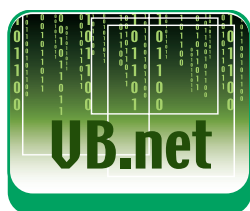
Quando si implementa un visualizer, occorre considerare che stiamo lavorando con il codice proveniente da due processi diversi: il codice del debugger (debugger side, che è il visualizer) ed il codice che invece viene debuggato (debuggee side, che è la nostra applicazione). Capire questa separazione è molto importante, perchè dobbiamo sempre tener presente che gli oggetti nel visualizer sono delle copie rispetto a quelle che girano nell'applicazione debuggata. Questo è il motivo principale per cui usiamo il metodo ReplaceObject, che internamente utilizza la serializzazione.

CONCLUSIONI

Sono ormai lontani i tempi in cui creare un software significava aver a che fare con semplici variabili di tipo int, float o string. Oggi la OOP dà enormi potenzialità, ma d'altro canto ha aumentato significativamente la complessità degli oggetti con i quali lavoriamo. I visualizer di Visual Studio 2005 ci permettono di debuggare intuitivamente e con più semplicità il nostro codice.

Liborio Igor Damiani

IL NAMESPACE MY SEMPLIFICA LA VITA

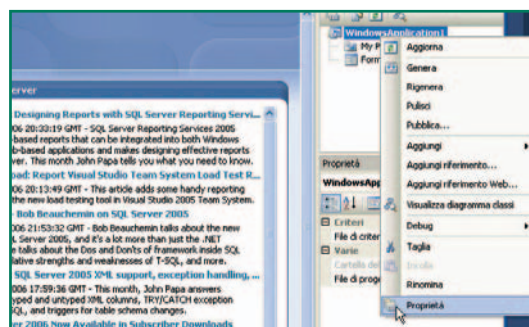
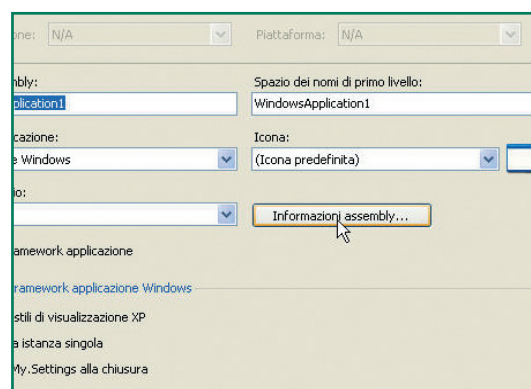
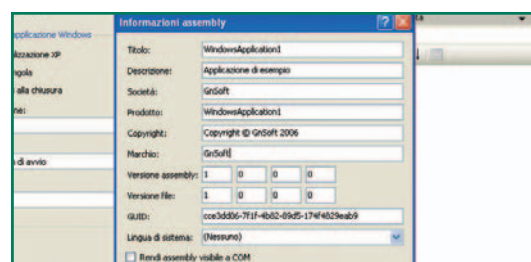
IN QUESTO ARTICOLO ILLUSTREREMO ALCUNE CARATTERISTICHE DEL NUOVO FRAMEWORK 2.0 CHE CONSENTONO DI ACCEDERE RAPIDAMENTE A INFORMAZIONI DI SISTEMA O ALTRI PERCORSI DI PROGETTO, SENZA TROPPE COMPLICAZIONI...

Microsoft Visual Basic 2005 introduce il nuovo namespace My, grazie al quale lo sviluppatore può accedere in modo rapido ad un gruppo di classi importanti relative a: il computer su cui l’applicazione è in esecuzione, l’utente che la sta eseguendo, l’applicazione stessa, i form dell’applicazione ed a qualsiasi Web service associato. Nel precedente articolo abbiamo analizzato in dettaglio l’oggetto **My.Computer** che permette di accedere alle informazioni relative al computer sul quale è installata l'applicazione, in questo articolo porteremo a termine la trattazione descrivendo alcuni altri oggetti messi a disposizione dal namespace My.

L’OGGETTO MY.APPLICATION

L’oggetto **My.Application** fornisce informazioni sull'applicazione in esecuzione, come il percorso, il titolo, la versione, la nazionalità e la modalità di autenticazione dell'utente.
L’oggetto **My.Application** espone un buon numero di proprietà e metodi.
Analizziamone alcuni.

L’oggetto Info, permette di ottenere versione, percorso, titolo, descrizione, working set, ed altre informazioni sull’assembly dell’applicazione. La maggior parte delle informazioni restituite dall’oggetto Info, sono raggiungibili per mezzo della finestra di dialogo: Informazioni assembly

UN ESEMPIO DI DIALOGO

1 Selezioniamo il nostro progetto di prova, nella finestra Esplora soluzioni e dal menu a tendina che si ottiene cliccando con il tasto destro del mouse, sulla nostra applicazione, selezioniamo la voce: Proprietà

2 A questo punto, viene visualizzata la finestra delle proprietà con la scheda di default Applicazione. Nella scheda Applicazione clicchiamo sul pulsante Informazioni assembly...

3 Così facendo, viene visualizzata la finestra di dialogo Informazioni assembly con le informazioni relative all’applicazione

Le informazioni esposte dall’oggetto Info, tipicamente si visualizzano nella classica finestra di dialogo about delle applicazioni Windows Form. Il codice seguente mostra in un TextBox una serie di informazioni ricavabili dall’oggetto Info:

```
Dim Informazioni As String = ""

'Visualizza il nome dell'applicazione.
Informazioni = Informazioni & "Nome:" &
        My.Application.Info.AssemblyName & vbCrLf

'Permette di visualizzare il nome della società
'associato all'applicazione.
Informazioni = Informazioni & "Società:" &
        My.Application.Info.CompanyName & vbCrLf

'Permette di visualizzare le informazioni sul
'copyright associate all'applicazione.
Informazioni = Informazioni & "Copyright:" &
        My.Application.Info.Copyright & vbCrLf

'Permette di visualizzare la descrizione associata
'all'applicazione.
Informazioni = Informazioni & "Descrizione:" &
        My.Application.Info.Description & vbCrLf

'Permette di visualizzare la directory in cui è
'memorizzata l'applicazione.
Informazioni = Informazioni & "Path:" &
        My.Application.Info.DirectoryPath & vbCrLf

'Permette di visualizzare il nome del prodotto
'associato all'applicazione.
Informazioni = Informazioni & "Prodotto:" &
        My.Application.Info.ProductName & vbCrLf

'Permette di visualizzare le informazioni relative
'all'analisi dello stack corrente.
Informazioni = Informazioni & "Stack:" &
        My.Application.Info.StackTrace & vbCrLf

'Permette di visualizzare il titolo associato
'all'applicazione.
Informazioni = Informazioni & "Titolo:" &
        My.Application.Info.Title & vbCrLf

'Permette di visualizzare le
' informazioni sul marchio
'associate all'applicazione.
Informazioni = Informazioni & "Marchio:" &
        My.Application.Info.Trademark & vbCrLf

'Permette di visualizzare il numero
'di versione dell'applicazione.
Informazioni = Informazioni & "Versione:" &
        My.Application.Info.Version.ToString & vbCrLf
'Permette di visualizzare la quantità di memoria
' fisica associata al contesto del processo.
Informazioni = Informazioni & "Memoria:" &
        My.Application.Info.WorkingSet & vbCrLf
TextBox1.Text = Informazioni
```

L’oggetto CommandLineArgs, permette di ottenere una ReadOnlyCollection di stringhe che racchiude gli argomenti della riga di comando dell'applicazione corrente.
Il codice seguente mostra in un TextBox, gli eventuali argomenti della riga di comando separati da punto e virgola (;)

```
Dim s As String = ""
Dim Informazioni As String = ""
For Each s In
    My.Application.CommandLineArgs
    Informazioni = Informazioni & s & ";"
Next
TextBox1.Text = Informazioni
```

L’oggetto Culture permette di ottenere la lingua (cultura) utilizzata dal thread corrente, il che influenza aspetti come l'elaborazione e la formattazione delle stringhe.
Per cambiare la lingua, possiamo utilizzare il metodo My.Application.ChangeCulture

GESTIAMO LA LINGUA

In questo esempio vediamo, in che modo, la lingua influenzi la rappresentazione in forma di stringa delle date. Definiamo la variabile che dovrà contenere la lingua corrente utilizzata nel sistema, e la inizializziamo

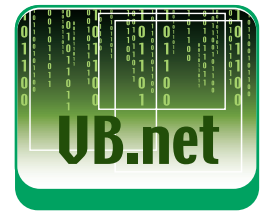
```
Dim LinguaCorrente As String =
        My.Application.Culture.Name
```

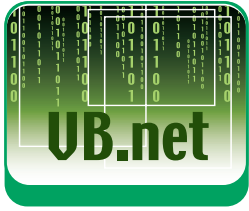
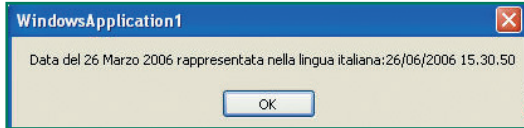
Visualizziamo un messaggio che indica la lingua corrente (nel nostro caso l’italiano)

```
MessageBox.Show ("La lingua utilizzata nel
        tuo sistema è: " & LinguaCorrente)
```

Definiamo la variabile che dovrà contenere la data di test, e la inizializziamo al 26 Marzo 2006 Ore 15:30 e 50 secondi.

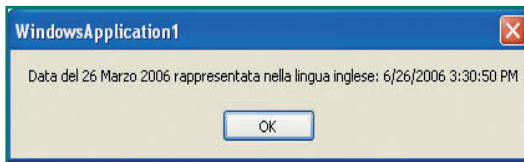
```
Dim DataDiProva As New Date(2006, 6, 26, 15,
        30, 50)
```

Visualizziamo un messaggio che mostra la data di test con le impostazioni correnti (abbiamo impostato come lingua di default l’italiano)

GESTIRE LE IMMAGINI

Le risorse immagine si possono assegnare alla proprietà Image di un controllo PictureBox in fase di progettazione senza scrivere una riga di codice. E’ sufficiente passare alla finestra delle proprietà, cliccare il pulsante con i tre puntini e selezionare la risorsa tra quelle che si è definito nella pagina Risorse del designer.

```
MessageBox.Show ("
        Data del 26 Marzo 2006
        rappresentata nella lingua italiana:" &
                                    DataDiProva)
```

Verrà visualizzato: 26/06/2006 15.30.50
Cambiamo la lingua in Inglese

```
My.Application.ChangeCulture("en-US")
```

Visualizziamo un messaggio che mostra la data di test con le impostazioni in inglese

```
MessageBox.Show (
 "Data del 26 Marzo 2006
        rappresentata nella lingua inglese: " &
                                    DataDiProva)
```

Verrà visualizzato: 6/26/2006 3:30:50 PM
Ripristiniamo la lingua iniziale

```
My.Application.ChangeCulture(LinguaCorrente)
```

NOTA

NAMESPACE ADATTABILI

Una caratteristica del namespace My è che non tutti i suoi oggetti vengono creati per tutti i tipi di progetti. Ad esempio, l'oggetto My.Form è disponibile solo nelle applicazioni Windows Forms. Il namespace My espone alcuni oggetti che vengono creati dinamicamente ogni volta che si aggiungono caratteristiche al progetto corrente.

L'OGGETTO MY.FORMS

L'oggetto My.Forms è subito disponibile, e definisce un metodo factory per ciascuna classe form definita nel progetto corrente. Le sue proprietà permettono di referenziare l'istanza di default di ciascuna Windows Form dichiarata nel progetto corrente, senza dover creare esplicitamente un oggetto della classe form. L'oggetto **My.Forms** da accesso ad una collezione che contiene i form dell'applicazione e permette di accedere ai form come si faceva in VB6.
Per i nostalgici, ricordiamo come in VB6, per mostrare uno dei form del progetto si doveva scrivere:

```
form1.Show
```

Con l'avvento del .NET Framework 1.0/1.1 non è stato più possibile usare la stessa sintassi ma si doveva scrivere:

```
Dim frm1 as New form1()
Frm1.Show
```

Con buona pace dei puristi della programmazione ad oggetti, che con l'avvento della versione 2.0 del Framework, si trovano a storcere il naso perché è possibile scrivere la seguente riga di codice equivalente al vecchio VB6

```
My.Forms.form1.Show
```

Possiamo notare, come il nome della proprietà sia uguale a quello del form a cui si accede, ed il tipo di proprietà equivale a quello del tipo di form.
Ciascuna proprietà dell'oggetto My.Forms fornisce accesso a un'istanza di un form nel progetto corrente. Per rimuovere un form, si deve assegnare il valore Nothing alla relativa proprietà. Se alla proprietà si assegna un valore diverso da Nothing, il metodo per l'impostazione genera un'eccezione Argument Exception.

L'OGGETTO MY.RESOURCES

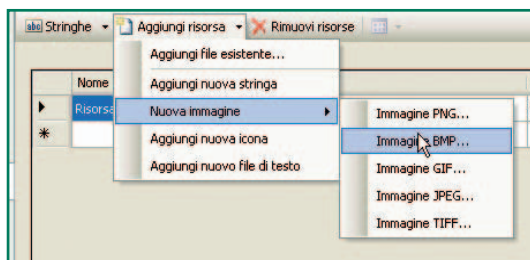
My.Resources contiene un oggetto per ciascuna risorsa definita nel progetto corrente. Se, ad esempio, aggiungiamo al progetto una bitmap denominata EdMaster, sarà possibile accedervi mediante My.Resources.EdMaster. Le risorse offrono un utile strumento per racchiudere in un assembly informazioni che diversamente dovrebbero essere fornite come file separati. Ad esempio si possono racchiudere un file di testo ed un file sonoro come risorsa, in questo modo si può accedere al testo ed al suono contenuti in questi file, senza doverli distribuire separatamente.

AGGIUNGERE RISORSE

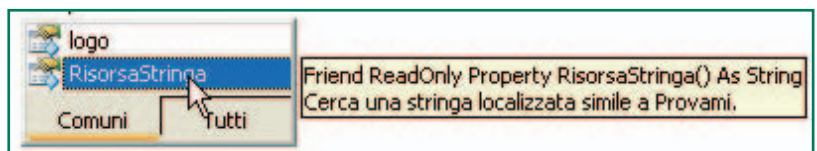
1 Selezioniamo il nostro progetto di prova, nella finestra: Esplora soluzioni e dal menu a tendina che si ottiene cliccando con il tasto destro del mouse, sulla nostra applicazione, selezioniamo la voce: Proprietà

2 A questo punto, viene visualizzata la finestra delle proprietà con la scheda di default Applicazione. Per Cambiare scheda clicchiamo sulla scheda Risorse

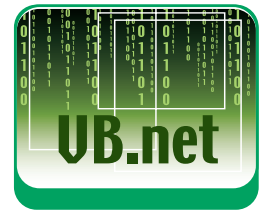
3 Aggiungere una risorsa di tipo stringa è molto semplice, è sufficiente inserire un nome di risorsa, il valore della stringa, ed un commento opzionale.

4 Per inserire altri tipi di risorse, è sufficiente trascinare un file esistente da Windows Explorer nella scheda Risorse. E’, inoltre, possibile creare una risorsa per mezzo degli editor disponibili in Visual Studio. Se ad esempio vogliamo creare un’immagine come risorsa, dobbiamo cliccare sul pulsante Aggiungi Risorsa. Verrà mostrato un menù a discesa da cui possiamo selezionare la voce Nuova immagine ed il tipo di immagine da creare.

Ogni volta che si aggiunge una nuova risorsa, nella scheda Risorse, queste vengono memorizzate nella cartella Resources della cartella del progetto (escluse le risorse stringa) e VB.NET popola l’oggetto My.Resources con il tipo corrispondente. Si possono aggiungere stringhe, immagini, icone, file .wav, ed ogni altro tipo di file.
Ad ogni risorsa deve essere assegnato un nome, e questo nome deve essere un identificativo valido di VB. Questo vincolo assicura che la risorsa possa essere esposta come proprietà dell'oggetto My.Resources. Le proprietà dell'oggetto My.Resources permettono di accedere in sola lettura alle risorse dell'applicazione.
Se, ad esempio vogliamo accedere alla risorsa di tipo stringa (RisorsaStringa) definita in precedenza, e visualizzarne il valore, possiamo scrivere:

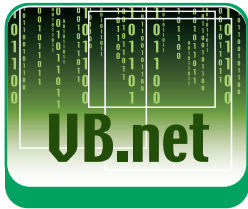
```
Dim EsempioRisorsa As String =
                        My.Resources.RisorsaStringa
MessageBox.Show(EsempioRisorsa)
```

Possiamo notare come l’intellisense, dopo il punto, ci mostra i nomi di tutte le risorse disponibili

Riassumendo, per ogni risorsa devono essere indicati: un nome, una categoria ed un valore. Il nome equivale al nome della proprietà, i dati della risorsa equivalgono al valore della proprietà, La categoria equivale al tipo della proprietà:

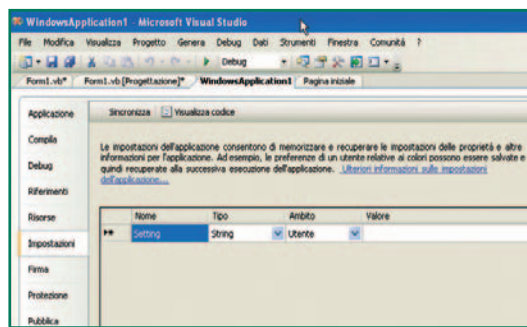
L’OGGETTO MY.SETTINGS

My.Settings mette a disposizione una proprietà per ciascuna impostazione di configurazione determinata nelle impostazioni correnti; a livello di applicazione o per utente. Una risorsa a livello di applicazione può essere condivisa da tutti gli utenti, una risorsa a livello di utente può essere modificata dall'utente corrente.
VB .NET memorizza le impostazioni in un file di configurazione XML, il cui nome corrisponde al nome del file eseguibile seguito dall’estensione .config. Nel nostro caso: WindowsApplication 1.exe.config In Visual Studio 2005 questi file di configurazione sono diventati più elastici e permettono di memorizzare impostazioni a livello di utente o di applicazione. Permettono, inoltre, di salvare le impostazioni quando l’applicazione termina. Le proprietà dell'oggetto My.Settings consentono di accedere alle impostazioni dell'applicazione. Ad ogni impostazione sono associati: Nome, Tipo, Ambito, Valore:

NOTA

PERCORSI SEMPLIFICATI

Il namespace My contiene un insieme di oggetti che forniscono percorsi semplificati a molte aree del .NET Framework, riducendo il numero di righe di codice che è necessario scrivere in maniera efficiente ed affidabile.

- Nome, permette di determinare il nome della proprietà.
- Tipo, permette di determinare il tipo della proprietà.
- Ambito, indica se la proprietà è in sola lettura. Se il valore è Applicazione, la proprietà è in sola lettura; se il valore è Utente, la proprietà è in lettura/scrittura.
- Valore, rappresenta il valore predefinito della proprietà.

AGGIUNGERE UNA STRINGA DI CONNESSIONE A DATABASE

1 Sempre dal nostro progetto di prova, nella finestra: Esplora soluzioni, selezioniamo la voce: Proprietà dal menu a tendina. Visualizziamo la scheda Impostazioni

2 Scriviamo il nome identificativo dell’impostazione, nella colonna Nome

3 Clicchiamo sull’elenco a discesa nella colonna Tipo e selezioniamo la voce (stringa di connessione)

4 L’ambito di validità diventa automaticamente a livello di applicazione

5 Infine clicchiamo il pulsante con i tre puntini (...) nella colonna valore, apparirà la finestra di dialogo: Scegli origine dati, da cui si può scegliere anche una fonte dati ODBC

My.Settings espone inoltre i due Metodi

- *Reload*, che permette di ricaricare le impostazioni dell'utente dagli ultimi valori salvati.
- *Save*, che permette di salvare le impostazioni correnti dell'utente.

CONCLUSIONI

Il namespace My semplifica moltissimo la vita al programmatore. Il suo utilizzo e semplice e immediato e le potenzialità enormi. Fatene buon uso.

Luigi Buono

NOTA

USARE L’INTELLISENSE

Gli oggetti del namespace My sono accessibili in modalità a tipizzazione forte, con l'aiuto di IntelliSense eliminando gli errori a runtime provocati da errori di digitazione.

USO DEGLI OPERATORI IS E ISNOT

Generalmente, l'operatore Is o IsNot deve leggere il valore della proprietà per eseguire il confronto. Tuttavia, se il valore della proprietà è Nothing, la proprietà crea una nuova istanza del form e quindi la restituisce. Il compilatore di Visual Basic considera le proprietà dell'oggetto My.Forms in modo speciale e consente all'operatore Is o IsNot di controllare lo stato della proprietà senza modificarne il valore.

SNMP: SISTEMA SOTTO CONTROLLO

JAVA DISPONE DI ALCUNE CLASSI CHE CI CONSENTONO DI INTERFACCIARE LE NOSTRE APPLICAZIONI CON SNMP. UTILIZZANDO QUESTO PROTOCOLLO È POSSIBILE TENERE TRACCIA DELL'ATTIVITÀ DI TUTTE LE RISORSE DEL COMPUTER. VEDIAMO COME...

Durante l'estate capita più spesso di divertirsi con i giochi enigmistici come, ad esempio, quello di trovare le differenze o le uguaglianze tra due foto. Nella figura 1 ci sono 2 valori uguali tra quelli presenti nel Task Manager e tra i risultati del programma. Come gioco non è molto difficile, come risultato invece è decisamente più interessante: il programma ha trovato che ci sono 56 processi in esecuzione. Volendo, in modo analogo, si possono determinare i valori della memoria o la descrizione del computer, l'ultima volta che è stato riavviato o la velocità della scheda di rete, o maggiori dettagli sui processi attivi con i loro identificativi e le porte sulle quali girano. E molto, molto altro! Questi parametri, inoltre, sono legati al funzionamento del computer e possono aprire molteplici scenari.
Basti pensare agli allarmi da generare se un certo processo non è più attivo, o se i livelli di memoria libera scendono sotto una determinata soglia. Infatti, non di rado, capita di guardare nelle sale server, persone che, appena vedono un valore anomalo, riavviano un determinato servizio o un programma.
Se questi valori del computer diventassero disponibili a livello di codice, si potrebbero anche intraprendere correzioni specifiche, come lanciare automaticamente un processo batch che liberi memoria o rilanci la procedura che sta assorbendo troppe risorse, senza che ci sia materialmente qualcuno a farlo. L'obiettivo di questo articolo è di costruire una applicazione, semplice da realizzare, che consenta di ricavare tutti quei parametri di sistema che permettono di tenere sotto controllo il computer, impiegando delle Api molto intuitive da usare, dando a tutti il controllo completo sui computer.
Tornando alla figura 1, si può notare come ci siano una serie di coppie di valori. A sinistra ci sono degli identificativi e a destra i corrispettivi valori. Il primo passo da compiere è capire il significato di questi

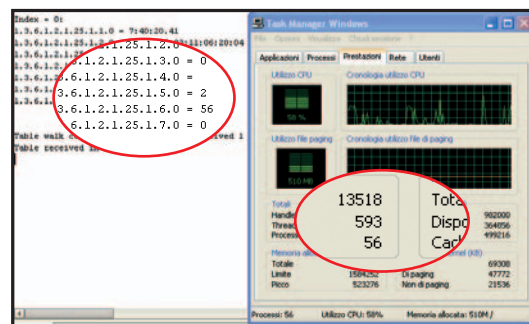



Fig. 1: **Esempio dell'applicazione.**

identificativi e il protocollo che permette di interrogarli: in questo modo, infatti, si può comprendere dove possono essere lanciati i vari elementi dell'applicazione e perché sia così facile reperire i dati che interessano.

SNMP, SIMPLE NETWORK MANAGEMENT PROTOCOL

È il protocollo (standard di fatto) di comunicazione per la gestione e il controllo della rete. I suoi punti di forza sono la semplicità e la facilità di uso. E anche l'estendibilità, consentendo l'introduzione di funzionalità e parametri aggiuntivi specifici per i diversi dispositivi e periferiche. Gli elementi che costituiscono l'architettura, come si può vedere bene in **figura 2**, sono di 2 tipi: un manager e gli agenti. Gli agenti hanno il compito di memorizzare e gestire i dati che interessano e possono inviare messaggi al manager. Il manager, a sua volta, ha diverse interazioni con gli agenti che può interrogare per verificarne i valori, ottenerne le risposte, può impostare i parametri e ricevere i messaggi. Già da questa prima descrizione si può tracciare il parallelo con quanto riportato in figura 1. Un manager ha interrogato un agente che gli ha fornito le risposte circa i parametri chiesti. E sono proprio i parametri che risultano ancora non ben definiti: cosa sono quelle sequenze come 1.3.6.1.2.1.25.1? Queste serie di numeri sono le unità elementari per gestire le informazioni e cioè i MIB, managment information base. In poche paro-

le, a ogni sequenza numerica corrisponde un valore. Nell'esempio di prima, 1.3.6.1.2.1.25.1.6.0 è l'identificativo riconosciuto che corrisponde al numero di processi attivi sulla macchina. Per poter essere interrogati da tutti, i MIB devono essere univoci e universali, permettendo di richiedere informazioni su macchine diverse, con hardware e sistemi operativi differenti, e ottenere sempre risposte congruenti. Considerando che SNMP è molto utilizzato, ci sono già le librerie che lo implementano. In questo caso, si utilizzerà snmp4j, un insieme di classi Java che permettono di sviluppare le applicazioni in modo estremamente semplice. SNMP è un protocollo di

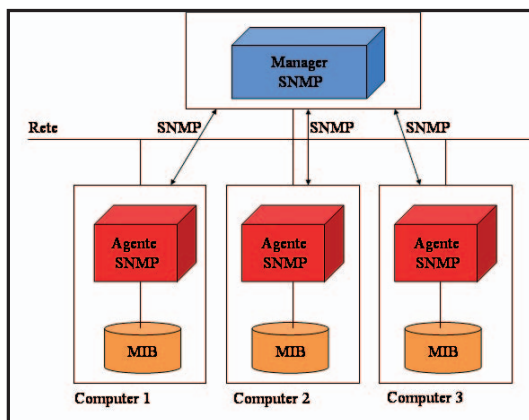



Fig. 2: Architettura SNMP

comunicazione tra manager e agenti che si scambiano messaggi. E sono proprio i messaggi che permettono di mandare informazioni sui parametri e sullo stato del computer.

NUMERO DI PROCESSI

L'esempio iniziale determinava quanti processi sono attivi sulla macchina. Per ottenere questo risultato, si può usare il comando GET del protocollo SNMP. Questo comando consente di ricavare il valore di un parametro specifico. Naturalmente il grosso del lavoro è svolto dalle api snmp4j che devono essere solo invocate correttamente. Quello che veramente serve sono 3 elementi: 1) la configurazione dei parametri relativi alla macchina, alla quale chiedere le informazioni, 2) la definizione del parametro da chiedere e 3) l'esecuzione dello scambio dei messaggi. Il codice mostra la semplicità di questi passi.

```
// Classe IoPGET
// configurazione Snmp
UdpAddress targetAddress = new
                    UdpAddress("127.0.0.1/161");
CommunityTarget target = new CommunityTarget();
target.setCommunity(new OctetString("public"));
target.setAddress(targetAddress);
```

Le istruzioni sopra servono per dire che si vogliono informazioni relative alla macchina locale, 127.0.0.1, e che si chiederanno sulla porta di default del protocollo UDP, la 161. Inoltre si imposta la comunità a public, per poter accedere ai parametri disponibili a tutti.

```
// preparare il PDU per l'invio messaggio
PDU pdu = new PDU();
pdu.setType(PDU.GET);
pdu.add(new VariableBinding(new
                    OID("1.3.6.1.2.1.25.1.6.0")));
```

Con queste direttive, si dichiara il comando da usare, GET, e il parametro per il quale si desidera conoscere il valore, 1.3.6.1.2.1.25.1.6.0. Le classi di snmp4j usano i nomi del protocollo. In particolare, OID, Object Identifier, è l'identificativo dell'oggetto caratterizzato dalla sequenza di numeri che attraversano l'albero e PDU, Protocol Data Unit, l'unita dei dati come ad esempio GET e GETNEXT.

```
Snmp snmp = new Snmp(new
                    DefaultUdpTransportMapping());
snmp.listen();
// invio del PDU
ResponseEvent responseEvent = snmp.send
                    (pdu, target);
if (responseEvent != null) {
  // gestione della risposta
  PDU pduResponse = responseEvent.getResponse();
  System.out.println("res: " +
                    pduResponse.toString());
...
```

A questo punto basta solo inoltrare la chiamata e aspettare la risposta. Il comando GETNEXT è simile e serve per prendere i valori successivi dell' OID.

```
pdu.setType(PDU.GETNEXT);
pdu.add(new VariableBinding(new
                    OID("1.3.6.1.2.1.1.5.0")));
```

Il commando GET restituisce il nome del computer, GETNEXT, invece, prende l'identificativo o gli identificativi successivi, ad esempio il 1.3.6.1.2.1.1.6.0 che corrisponde alla posizione della macchina

INVIARE I PARAMETRI

Un altro modo per usare le api snmp4j è lanciando la classe org.snmp4j.tools.console.Snmp Request con vari parametri come specificato nella classe stessa. Un esempio di parametri è : -p TRAP -v 3 -u aSecurityName 127.0.0.1/162 "1.3.6.1.2.1.1.3.0={t}0" "1.3.6.1.6.3.1.1.4.1.0={o}1.3.6.1.6.3.1.1.5.1" "1.3.6.1.2.1.1.1.0={s}System XYZ, Version N.M"

all'interno della rete. A questo punto ci si può divertire a prelevare tutte le informazioni che servono. Ad esempio il valore 1.3.6.1.2.1.1.1.0 serve per avere informazioni hardware e sistema operativo del computer, 1.3.6.1.2.1.1.3.0 indica da quanto tempo il sistema è attivo, 1.3.6.1.2.1.1.5.0 è il nome del computer, e così via. Ora serve solo un po' di ricerca per trovare gli identificativi che interessano. I comandi considerati prima servono per prendere i valori del

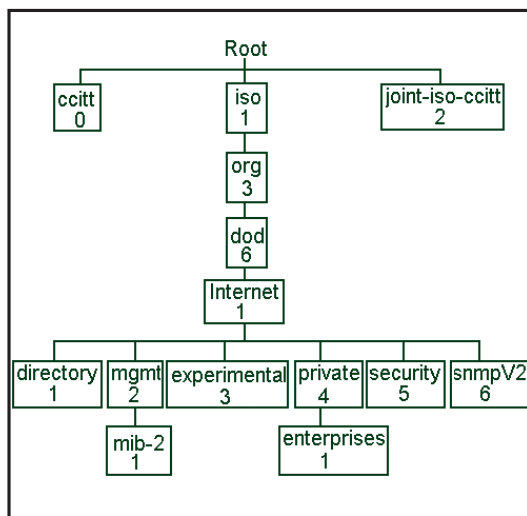



Fig. 3: **Gerarchia MIB.**

computer. Se invece si volesse impostare un valore, allora si può usare il comando SET. E con questo comando, termina l'insieme delle operazioni più dirette da compiere. In realtà si possono aprire scenari più interessanti.

LA FINE DEI PROBLEMI

Una situazione tipica per un server è che vi giri un'applicazione. E se l'applicazione, per qualche motivo, non dovesse più funzionare? E se assorbisse troppe risorse? Ritornando all'architettura di SNMP ci sono 2 attori, il manager e gli agenti. Implicitamente, nell'esempio di prima, si è assunto che il programma funzionasse da manager e interrogasse un agente chiedendogli i parametri. In realtà ci si può mettere nei panni dell'agente che gira sul server il quale, accorgendosi, ad esempio, che un processo non è più attivo, invia una segnalazione al manager. In questo caso si dovranno realizzare sia il manager, che resta in ascolto di possibili problemi dell'agente, sia l'agente che invia la comunicazione. Generalmente la parte più importante spetta al manager che, in base all'allarme generato, compie le modifiche necessarie. Anche in questo caso servono poche righe di codice per catturare quella che nel protocollo viene definita TRAP.

```
// Classe IoPTrapListener
// gestione della trap
  CommandResponder trapPrinter = new
                              CommandResponder() {
    public synchronized void
          processPdu(CommandResponderEvent e) {
      PDU command = e.getPDU();
      if (command != null) {
        System.out.println(command.toString());
        // gestione specifica...
      }
    }
  };
  snmp.addCommandResponder(trapPrinter);
```

Da notare anche la gestione di thread concorrenti grazie all'uso del syncronized. In questo modo il manager è in ascolto. Dall'altro lato, invece, un agente invia la TRAP.

```
PDUv1 pdu = new PDUv1();
pdu.setType(PDU.V1TRAP);
pdu.setGenericTrap(PDUv1.COLDSTART);
```

Per mandare le TRAP si possono usare varie classi, a seconda della versione del protocollo. La classe base è PDU che imposta il proprio tipo usando .setType(PDU.TRAP). Invece, per la versione 1 di SNMP si può usare la classe PDUv1 che è derivata da PDU. L'ulteriore parametro, COLDSTART, tiene conto delle possibili modifiche che possono influire sulla configurazione dell'agente.

SICUREZZA

Dagli esempi e dall'architettura si nota che SNMP può funzionare sia in locale, per controllare la propria macchina e prendere i parametri, sia, soprattutto, in contesti distribuiti. Uno scenario tipico è un manager installato su una macchina e vari agent sui computer da tenere sotto controllo che si scambiano messaggi. Inutile dire che a molti viene l'acquolina in bocca solo a pensare come fare a inserirsi nello scambio di messaggi, a contraffarli, a provare a mandare informazioni sbagliate. Ed effettivamente, se a fronte di un messaggio di TRAP lanciato da un agent, il manager decidesse di riavviare un servizio, a un hacker basterebbe lanciare TRAP una dopo l'altra

USARE SNMP4J DA CONSOLE

Un altro esempio dell'utilizzo delle api snmp4j è realizzato dalla console snmp4j-agent presente anch'essa nel file allegato. Può funzionare come applicazione a sé e, grazie al codice sorgente, può dare intere righe di codice funzionante.

per mandare giù tutto il sistema. Al di là dell'esempio, e se ne potrebbero fare molti altri, il problema della sicurezza è decisivo anche in questo caso. SNMP utilizza due elementi per validare una macchina che vuole scambiare messaggi: il nome della comunità e l'indirizzo IP. In particolare, quello che è possibile gestire via codice è la comunità e per farlo servono due righe di codice.

```
CommunityTarget target = new CommunityTarget();
target.setCommunity(new OctetString("public"));
```

Già in questo modo si ha una certa garanzia di concedere solo i privilegi necessari. Ricordando, infatti, i parametri che si possono ottenere da una macchina, alcuni di essi possono essere letti e modificati, altri è necessario solo leggerli senza dare la possibilità di modifica. Ad esempio si può voler modificare i parametri di timeout, mentre non avrebbe senso cambiare l'identificativo, univoco, delle interfacce di rete. Un ulteriore livello di sicurezza può essere impostato via codice usando le api snmp4j e la crittografia. Questa volta le righe di codice sono un po' di più perché gli elementi in causa sono più numerosi: il protocollo, l'utente, il destinatario e il sistema di crittografia. Il codice è comunque abbastanza auto esplicativo.

```
// Classe IoPAuth
…
MPv3 mpv3 =
   (MPv3)
snmp.getMessageProcessingModel(MessageProcessing
                                     Model.MPv3);
USM usm = new
                  USM(SecurityProtocols.getInstance(),
    new OctetString(mpv3.createLocalEngineID()), 0);
SecurityModels.getInstance().addSecurityModel(usm
                                                 ;
 transport.listen();
// aggiunta dello user all' USM
snmp.getUSM().addUser(new
                              OctetString("MD5DES"),
   new UsmUser(new OctetString("MD5DES"),
      AuthMD5.ID,
      new OctetString("MD5DESUserAuthPassword"),
      PrivDES.ID,
      new OctetString("MD5DESUserPrivPassword")));
// creazione del targhet
UserTarget target = new UserTarget();
…
target.setVersion(SnmpConstants.version3);
target.setSecurityLevel(SecurityLevel.AUTH_PRIV);
target.setSecurityName(new
                             OctetString("MD5DES"));
…
```

Lo scambio dei messaggi dipende dal protocollo che si usa e, in questo caso, è la versione 3 di SNMP. Vi è poi la classe relativa all'utente, USM, User Based Security Model, che ne gestisce anche le credenziali. Le ultime righe servono per impostare i parametri dei messaggi e della sicurezza anche sulla macchina targhet, quella alla quale mandare i messaggi. I protocolli di crittografia che si possono usare dalle api sono MD5, DES, come nell'esempio, e anche AES e SHA. In poche parole sono rappresentati tutti modi usati attualmente per rendere sicuro lo scambio di informazioni in rete. Un'altra istruzione utile è relativa al livello di sicurezza. La classe SecurityLevel definisce i livelli di sicurezza che possono essere usati. Entrando nel codice ci sono dichiarate 3 costanti che sprigionano scenari assai diversi tra di loro.

```
// Classe SecurityLevel
public static final int NOAUTH_NOPRIV = 1;
public static final int AUTH_NOPRIV = 2;
public static final int AUTH_PRIV = 3;
```

Anche i nomi sono di facile interpretazione. Nel primo caso, chiunque può creare e leggere messaggi. L'altro caso estremo è l'ultimo dove solo chi ha la chiave per autenticarsi può creare messaggi e solo chi ha le chiavi per criptare e decriptare può leggere il contenuto dei messaggi. Il secondo caso è quello intermedio dove per creare i messaggi serve la chiave e tutti possono leggerli. In questo ultimo caso c'è

CHE COSA È UN MIB?

Per sapere cosa è esattamente un mib e vedere tutti parametri che si possono usare, basta ricercare, sulla macchina, i file con estensione .mib. All'interno si possono leggere le definizioni dei mib e il codice numerico associato ai parametri.

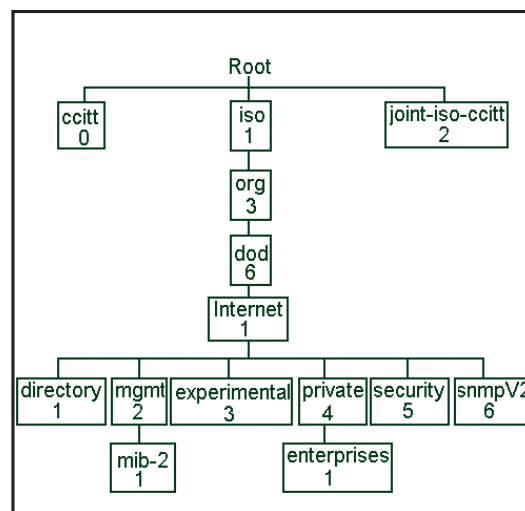



Fig. 4: ***Flusso dei messaggi***

una asimmetria con sicurezza solo su chi crea il messaggio che, in generale, è anche la parte più delicata.

MESSAGGI

Dopo aver visto che sono veramente pochi i tipi di messaggio che si possono inviare e aver considerato i problemi di sicurezza, si può passare a un altro dei problemi che coinvolge la rete: lo scambio dei messaggi. Lo scambio dei messaggi avviene in 2 modi: sincrono e asincrono; il primo caso è come l' invocazione di una pagina web con un browser che fa una richiesta a un server e resta in attesa finché il server gli restituisce la pagina desiderata oppure si verifica un timeout. Il secondo caso è come quello della posta elettronica: la posta viene inviata da un utente che non resta in attesa finché il destinatario non la legge. La differenza principale tra queste due modalità è nel fatto che un messaggio sia bloccante o meno e cioè se il mittente resta in attesa di risposta o può scollegarsi appena inviato il messaggio.
Considerando le velocità e la garanzia dei servizi, spesso non è possibile aspettare 2 minuti prima di ricevere un timeout e nel frattempo bloccare tutti gli altri messaggi. Ecco quindi che, se si vogliono sfruttare appieno le caratteristiche di snmp4j, si dovrà tenere conto anche di come inviare e ricevere i messaggi. Anche in questo caso le operazioni da compiere sono poche e semplici e avranno poi impatti importanti ai fini delle prestazioni. Un esempio di messaggio sincrono che aspetta una risposta immediata è il seguente.

```
Snmp snmp = new Snmp(new
                    DefaultUdpTransportMapping());
...
ResponseEvent response = snmp.send(requestPDU,
                                        target);
```

Gruppo	Parametro	Descrizione
System	1.3.6.1.2.1.1.1.0	Descrizione testuale della macchina
System	1.3.6.1.2.1.1.3.0	Tempo dall'ultimo riavvio
System	1.3.6.1.2.1.1.6.0	Posizione fisica
Interfacce	1.3.6.1.2.2.1.2.ifIndex	Descrizione dell'interfaccia
Interfacce	1.3.6.1.2.1.2.2.1.4.ifIndex	Dimensioni massime pacchetto
IP	1.3.6.1.2.1.4.2.0	Valore Ip
IP	1.3.6.1.2.1.4.3.0	Numero pacchetti ricevuti
TCP	1.3.6.1.2.1.1.6.10.0	Numero pacchetti ricevuti
TCP	1.3.6.1.2.1.6.12.0	Numero segmenti ritrasmessi
UDP	1.3.6.1.2.1.7.5.1.1	Indirizzo IP locale
SNMP	1.3.6.1.2.1.11.25.0	Numero di GET
SNMP	1.3.6.1.2.1.11.30.0	Permesso di generare trap
Vari	1.3.6.1.2.1.25.2.2.0	Dimensione totale memoria
Vari	1.3.6.1.4.1.2.6.191.2.4.2.1.4.1	Dimensione pagine di swap
Vari	1.3.6.1.2.1.25.2.3.1.5.8	Memoria libera
Vari	1.3.6.1.4.1.17384.1.1.1	Temperatura

```
if (response.getResponse() == null) {
...
```

Il caso di messaggio sincrono si basa sull'uso della classe ResponseEvent nella quale viene scritto anche l'esito del messaggio. Anche il codice per mandare messaggi asincroni è facile.

```
Snmp snmp = new Snmp(new
                    DefaultUdpTransportMapping());
...
ResponseListener listener = new ResponseListener()
                                                  {
  public void onResponse(ResponseEvent event) {
    PDU response = event.getResponse();
    PDU request = event.getRequest();
    if (response == null) {
      System.out.println("Request "+request+"
                                    timed out");
    }
    else {
      System.out.println("Received response
                  "+response+" on request "+
                  request);
    }
};
snmp.sendPDU(request, target, null, listener);
...
```

In questo caso è la classe ResponseListener che invia messaggi in modalità asincrona usando una classe interna e l'oggetto PDU per prendere i dati. Questa è la struttura di chi invia il messaggio. In modo speculare, chi riceve il messaggio deve porsi in ascolto.

```
snmp.addCommandResponder(this);
transport.listen();
...
public synchronized void
        processPdu(CommandResponderEvent e) {
  PDU command = e.getPDU();
  if (command != null) {
  ...
  }
}
```

MIB

Usare SNMP ha senso se ci sono dei parametri che interessano. E in realtà i parametri sono davvero molti e seguono una logica ben precisa. Come si può vedere dalle figure 3 e 5, i primi valori, 1.3.6.1, sono uguali per molti dei parametri principali. Sotto al nodo che caratterizza internet, i due valori più comuni sono 4, private, e 2, mgmt. Sotto mgmt ci sono i parametri del MIB2 e sono i più noti. Il gruppo 1, system, ha delle informazioni sul sistema come

Fig. 5: ***Gerarchia MIB II***

la descrizione del sistema operativo, e il tempo trascorso dall'ultimo riavvio. Il gruppo delle interfacce di rete da informazioni sulle schede, e poi ci sono i gruppi per IP, ICMP, TCP, UDP, SNMP e errori. Dare una descrizione completa di tutti sarebbe molto lungo. Alcuni di questi parametri sono riportati in tabella.

STORIA

Dalle situazioni analizzate in precedenza, emerge prepotentemente la possibilità di verificare i parametri e, se sono errati, compiere modifiche correttive. Può essere utile, quindi, salvare tutti i dati nel database. In questo modo si può sempre accedere ai valori che aveva un determinato parametro nel corso del tempo. Ovvero si può ricavare la storia di un determinato parametro. Per poter effettuare queste operazioni, si deve organizzare le informazioni che si possono recuperare per poterle poi inserire in un database. Si può scegliere un qualsiasi database, anche gratuito, e il codice resta quasi invariato. Nel caso specifico il database di riferimento è Oracle. I passi da seguire sono essenzialmente 3: 1) recupero del parametro di interesse, 2) formattazione adeguata del parametro da inserire nel database, 3) inserimento nel database. Questo flusso viene gestito da una classe principale che a sua volta chiama le due classi specifiche per il recupero del dato e per la memorizzazione.

```
// Classe IoPStore
//recupero dati
IoPGET iopGET = new IoPGET();
String keyValue = iopGET.getKeyValue();

//formattazione dati
String key;
String value;
String[] splitted = keyValue.split("=");
key = splitted[0];
value = splitted[1];

//inserimento nel db
OracleDAO oracleDAO = new OracleDAO();
oracleDAO.insertValue(key, value);
```

Come si può esaminare dal codice, le operazioni sono semplici. La classe IoPGET è quella utilizzata negli esempi precedenti per recuperare i valori. Il dato che viene fornito è del tipo <chiave> = <valore> e quindi si rende necessaria un'operazione di split per dividere chiave e valore da passare poi alla classe OracleDAO che si occupa della memorizzazione.

```
//Classe OracleDAO
private Connection con;
private Statement stmt;
…
try {

   prepareConnection();
   stmt = con.createStatement();
   stmt.executeUpdate("INSERT INTO PARAM (IDKEY,
                              VALUE, RETRIEVETIME) " +
      "VALUES ('" + key + "', '" + value + "',
                                            SYSDATE)");
   stmt.close();
   con.close();
} catch (SQLException e) {…}
```

Per inserire il record serve solo prendere una connessione, inizializzarla con i parametri specifici del database ed eseguire lo statement.

CONFIGURAZIONI

Il protocollo SNMP è implementato da molti software che possono da operare sia da manager sia da agenti. Nei dischi di installazione sono già presenti, sia nelle varie versioni Windows sia Linux. Questi prodotti funzionano sulle porte udp standard, 161 e 162 e possono colloquiare con le classi che vengono realizzate a patto che queste usino le stesse porte. Viceversa, se si vogliono realizzare sia manager che agenti, si deve verificare che le porte usate siano libere. Comandi utili sono netstat e ps che visualizzano i processi attivi e le porte corrispondenti. In questo modo si può anche verificare se sono stati installati più programmi che usano il protocollo SNMP e quale di essi è effettivamente funzionante.

CONCLUSIONI

Il protocollo SNMP garantisce il controllo delle macchine e, con pochi e semplici comandi, permette tutta le gestione di intere reti. Le api snmp4j sono molto utili per chi voglia realizzare programmi personalizzati per la gestione del sistema. Essendo poi scritte in Java, funzionano su tutti i sistemi operativi e non serve software aggiuntivo per far comunicare e amministrare una rete di macchine di qualsiasi tipo.

Cristiano Bellucci

ARRIVA LA SHELL DI WINDOWS

GLI UTENTI UNIX SONO ABITUATI A LAVORARE A RIGA DI COMANDO. DOPO TANTI ANNI ANCHE MICROSOFT HA DECISO DI FORNIRE STRUMENTI AVANZATI DI AUTOMAZIONE PER LA GESTIONE DEL SISTEMA BASATI SU SHELL. VEDIAMO QUALI

In questi ultimi mesi si sente spesso parlare di un nuovo componente, che sarà rilasciato con la nuova versione del sistema operativo Windows, denominato Monad. Non è passato molto tempo, però, da quando la stessa Microsoft ha modificato ed ufficializzato il nome definitivo con cui intende indicare questo "nuovo" oggetto: Windows PowerShell.
Malgrado il termine PowerShell possa far pensare "semplicemente" ad una delle tante nuove utility di casa Microsoft, l'ex-Monad (com'eravamo "abituati" a chiamarla) o MSH, come vedremo, è certamente qualcosa di più di una semplice shell. In questo articolo passeremo in rassegna alcuni degli aspetti che ruotano attorno ad essa, sottolineando sin d'ora che quanto mostreremo è, in realtà, solo una minima parte di quello che la PowerShell è in grado di fare.

PREMESSA

Iniziamo subito questo percorso dicendo (e rassicurando tutti) che la tecnologia oggetto di questo articolo sarà integrata nativamente nel nuovo sistema operativo Windows Vista, ma che verrà resa disponibile anche su Windows XP e Windows Server 2003 mediante l'applicazione di una patch.
Al momento in cui scriviamo, è possibile cominciare a prendere visione del funzionamento di questa shell collegandosi al sito della Microsoft http://www.microsoft.com/downloads/details.aspx?FamilyId=2B0BBFCD-0797-4083-A817-5E6A054A85C9&displaylang=en e scaricando quanto necessario alla sua esecuzione.
La Windows PowerShell necessita dell'installazione del .NET Framework 2.0 RTM, scaricabile dal medesimo link ed indispensabile per il suo funzionamento. Per i test sono stati utilizzati due PC separati, il primo con Windows XP Home Ed. ed il secondo con Windows 2003 R2. In entrambi i casi, è bene sottolinearlo, non sono stati riscontrati particolari problemi di compatibilità o altro.

Per cominciare, a questo punto, partiamo subito da una piccola considerazione, dando per scontato che l'installazione della PowerShell e del .NET Framework sia terminata senza particolari problemi. Tutti quanti noi ricordiamo perfettamente il significato che avevano i file Autoexec.bat e Config.sys. Questi file erano utili per impostare determinati parametri all'avvio del computer o, comunque, in grado di preimpostare l'ambiente in base alle esigenze.
La Windows PowerShell consente di configurare il proprio ambiente in maniera analoga, servendosi di appositi file con estensione PS1 (la nuova estensione con cui vengono contrassegnati gli script PowerShell).
Dalla documentazione a corredo del pacchetto, si evince che all'avvio della shell vengono caricati diversi profili, nell'ordine seguente:

```
Documents and Settings\All
        Users\Documents\PSConfiguration\profile.ps1;
Documents and Settings\All
          Users\Documents\PSH\Microsoft.Power
                                    Shell_ profile.ps1;
$HOME\My Documents\PSConfiguration\
          profile.ps1;
$HOME\My
Documents\PSH\Microsoft.PowerShell_profile.ps1.
```

Ciascun file di profile serve a caricare impostazioni comuni e non a tutti gli utenti della shell. Onde evitare inutili perdite di tempo, si tenga presente che la dicitura $HOME equivale proprio alla variabile d'ambiente HOME preimpostata all'interno della PowerShell. Per vederne il contenuto è sufficiente lanciare il comando $HOME direttamente dalla linea di comando della shell. Un possibile valore, ad esempio, potrebbe essere *C:\Document and Settings\Francesco*.
Al momento sembrerebbe che l'installazione della PowerShell contenuta all'interno del pacchetto offerto dalla Microsoft non si preoccupi affatto di creare queste cartelle né tantomeno di

provvedere a file standard a cui fare riferimento. Per motivi di spazio, ma anche e soprattutto perché l'argomento può essere approfondito solo dopo una minima conoscenza di base sul nuovo linguaggio di scripting, non ci soffermeremo su questo "particolare". In ogni caso, quando saremo pronti, tutto quello che occorrerà fare, sarà creare le cartelle nell'ordine visto precedentemente ed, al loro interno, inserire i file di profilo necessari seguendo la nomenclatura vista prima.
Nel caso sia la prima volta che avviamo la PowerShell, ci verrà mostrato un messaggio importante che ci avverte sull'impossibilità di avviare script. Questo avvertimento, noto come Execution Policy, è ovviamente inteso a proteggere il sistema dall'esecuzione di codice non sicuro. Per modificare in qualunque momento le eventuali restrizioni impostate, è sufficiente lanciare il comando Set-ExecutionPolicy seguito dal parametro (numerico o testuale) corretto che ne identifica l'eventuale comportamento, secondo le indicazioni seguenti:

- UnRestricted (0): nessuna restrizione;
- RemoteSigned (1): tutti gli script scaricati da web devono avere una firma digitale di un proprietario garantito;
- Allsigned (2): tutti gli script devono avere una firma digitale per poter essere eseguiti;
- Restricted (3): nessuno script può essere avviato.

Per esempio, i due comandi seguenti sono assolutamente equivalenti:

```
Set-ExecutionPolicy UnRestricted
Set-ExecutionPolicy 0
```

```
Windows PowerShell
PS C:\Documents and Settings\Francesco\Desktop> get-help
TOPIC
    Get-Help

SHORT DESCRIPTION
    Displays help about PowerShell cmdlets and concepts.

LONG DESCRIPTION

SYNTAX
    get-help {<CmdletName> | <TopicName>}
    help {<CmdletName> | <TopicName>}
    <CmdletName> -?

    "Get-help" and "-?" display help on one page.
    "Help" displays help on multiple pages.

    Examples:
      get-help get-process   : Displays help about the get-process cmdlet.
      get-help about-signing : Displays help about the signing concept.
      help where-object      : Displays help about the where-object cmdlet.
      help about_foreach     : Displays help about foreach loops in PowerShell.
      match-string -?        : Displays help about the match-string cmdlet.

    You can use wildcard characters in the help commands (not with -?).
    If multiple help topics match, PowerShell displays a list of matching
    topics. If only one help topic matches, PowerShell displays the topic.

    Examples:
      get-help *       : Displays all help topics.
      get-help get-*   : Displays topics that begin with get-.
      help *object*    : Displays topics with "object" in the name.
      get-help about* : Displays all conceptual topics.
```

Fig. 1: ***Porzione dell'output prodotto dal cmdlet get-help***

Solo per curiosità, ecco la chiave del Registry sulla quale hanno impatto tali modifiche: //HKEY_LOCAL_MACHINE/SOFTWARE/Microsoft/PowerShell/1/ShellIds/Microsoft.PowerShell\ExecutionPolicy.
Prima di passare al concetto base della PowerShell, le Cmdlet, terminiamo queste poche righe menzionando subito il comando che ci permetterà, in qualunque momento, di ricavare informazioni d'aiuto sulla shell: get-help. Se lo lanciamo, otterremo a video una lunga descrizione che illustra i metodi con cui possiamo sfruttarlo. Inutile ribadire sin d'ora che, col passare del tempo, scopriremo molto presto di non poterne più fare a meno.

CENNI SUI CMDLET

Il concetto attorno al quale ruota l'utilizzo della nuova shell di casa Microsoft è certamente costituito da questi oggetti. Pronunciato "command let", questo termine identifica genericamente uno dei tanti comandi che possiamo utilizzare all'interno dell'ambiente offerto dalla Power Shell. Ad onor del vero, va sottolineato che la maggior parte delle cmdlet sono molto di più di un semplice comando ed avremo modo di rendercene conto con l'esperienza.
Innanzitutto, se lanciamo il comando get-command otteniamo la lista di tutti i cmdlet disponibili con accanto una parziale descrizione. L'elenco è piuttosto lungo (se ne contano 129). Alcuni di essi sono intuitivamente semplici da utilizzare, per altri invece, occorre documentarsi prima di capire realmente a cosa servano e soprattutto come utilizzarli.
Un cmdlet può essere lanciato allo stesso modo in cui eravamo abituati ad avviare qualunque comando in ambiente DOS.
Ecco a titolo d'esempio qualche cmdlet che può risultarci utile:

- get-service: produce una lista di tutti i servizi avviati e non sul sistema;
- get-process: produce una lista di tutti i processi attivi;
- get-alias: produce una lista di tutti gli alias definiti all'interno della shell, che permettono di riferirci ad un cmdlet attraverso un nome più corto.

Sarebbe davvero un'impresa descrivere nei dettagli tutti i cmdlet disponibili, soprattutto perché, anche considerando quelli "di poco conto", la maggior parte d'essi va studiato a fondo prima di poterne fare un utilizzo proficuo. Per non disperdere le nostre "energie" quindi, concentre-

remo la nostra attenzione su alcuni di essi. In seguito, nei paragrafi successivi, vedremo qualche altro Command-Let interessante.
Prendiamo il caso del get-service. Se lanciato, produce un risultato del tipo:

Status	Name	DisplayName
------	----	-----------
Stopped	Alerter	Avvisi
Running	ALG	Servizio Gateway di livello applica...
Stopped	AppMgmt	Gestione applicazione
...		

Lanciamo ora i seguenti comandi:

```
get-service | where { $_.status -eq "Running" }
get-service | where { $_.Name -eq "Alerter" }
```

Nel primo caso otterremo la lista dei soli servizi già avviati, mentre nel secondo solo i dettagli del servizio Alerter. Malgrado il secondo comando (ma anche il primo) si potessero ottenere in maniera più semplice (ad esempio, nel secondo caso bastava un get-service –Name "Alerter), da queste due righe possiamo desumere almeno un paio di cose.
La prima è l'utilizzo delle pipe (identificate attraverso l'utilizzo del carattere "|"). In questa maniera, l'output di un comando viene dato in pasto ad un altro e rielaborato. Sin qui, ovviamente, nulla di speciale. Ma consideriamo la porzione del comando

```
$_.status -eq "Running"
```

Il $_ identifica il generico oggetto (nel nostro caso, il nome di un servizio) corrente. In questa maniera, controllando la proprietà Status del servizio corrente ed assicurandoci che sia uguale a Running, il servizio viene aggiunto o meno nella lista finale. Questa "coppia di simboli" la incontreremo spesso negli script e pertanto è importante aver compreso bene a cosa si riferisca.
Per il controllo sullo stato del servizio è stato utilizzato il –eq che equivale all'operatore d'uguaglianza, uno dei più sfruttati tra quelli a disposizione. Analogamente a quanto poteva accadere con semplici comandi DOS, la maggior parte di quelli disponibili all'interno della PowerShell produce dei risultati tabellari (come quelli visti precedentemente). Spesso, però, questi risultati, per necessità o altro, devono essere trattati (perché magari ci occorre una lista più breve, con meno colonne, ecc.) e dati in pasto ad un altro cmdlet. Non sempre i cmdlet dispongono di switch in grado di fornire un risultato soddisfacente. In questi casi possiamo usare alcuni cmdlet adatti a questo scopo:

- Format-List
- Format-Custom
- Format-Table
- Format-Wide

Ad esempio:

```
get-service -Name "Lanmanserver"|Format-List
```

produce:

Name	: lanmanserver
DisplayName	: Server
Status	: Running
DependentServices :	{Browser}
ServicesDependedOn	: {}
CanPauseAndContinue	: True
CanShutdown	: True
CanStop	: True
ServiceType	: Win32ShareProcess

Mentre il comando:

```
Get-service | Format-Wide Name –column 3
```

produce:

Alerter	ALG	AppMgmt
aspnet_state	AudioSrv	awhost32
BITS	Browser	CiSvc
ClipSrv	clr_optimization_v2.0.5...	COMSysApp
...		

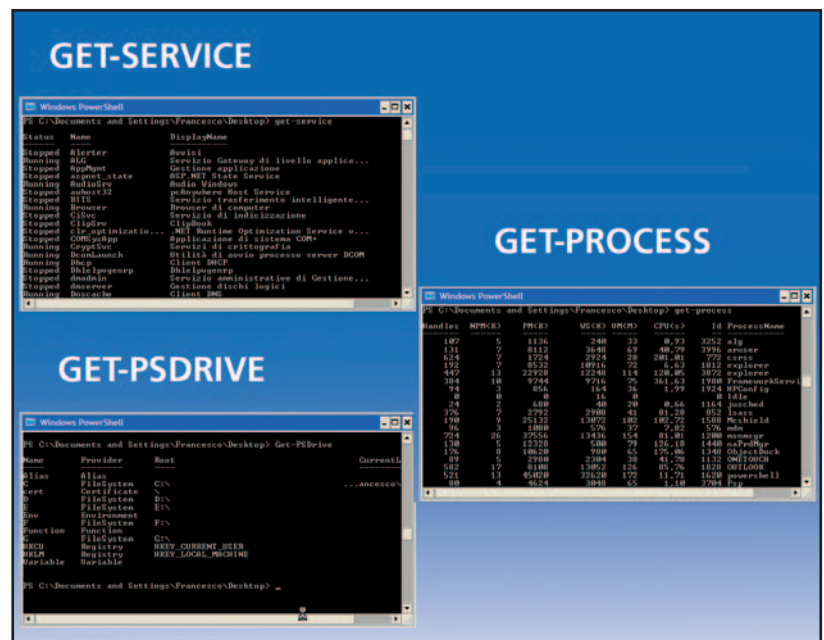



Fig. 2: I cmdlet get-service, get-process e get-alias in azione.

Occorre sempre fare un po' di pratica con questi strumenti, ma presto impareremo ad apprezzarne le potenzialità.

POWERSHELL SCRIPTING

Il punto di forza della nuova shell di casa Microsoft risiede certamente nella qualità e potenza dei cmdlet, che probabilmente non sarebbero "nulla" se non fossero affiancati da un altrettanto potente linguaggio di scripting. Per molti aspetti la sintassi delle istruzioni che possono essere utilizzate all'interno di questi nuovi "batch" ricorda molto un "miscuglio" tra C++ e comandi DOS, anche se, rispetto a quest'ultimi, sono molto più flessibili da utilizzare.

Non è certo questa la sede per poter elencare e spiegare tutte le istruzioni che possiamo utilizzare con i programmi della PowerShell, perché non basterebbero queste poche righe. Tuttavia, ci sembra utile mostrare uno script di media complessità dal quale evidenziare un po' delle caratteristiche di questi nuovi programmi.

Il codice seguente si preoccupa di recuperare tutte le informazioni sulla configurazione dei parametri legati alla rete. Anche se può sembrare molto complesso, più avanti dovrebbe risultarci più chiaro.

```
#----------------------------- #
#Nome file: Display_IP_Config.ps1 #
#----------------------------- #

#Raccolta parametri dal Registry
#(alcuni non sono utilizzati)
$TCPParametersKey="HKLM:\System\CurrentControl
                     et\Services\tcpip\parameters"
  $DataBasePath=$(Get-ItemProperty
                   $TCPParametersKey).DataBasePath
  $DeadGWDetectDefault=$(Get-ItemProperty
                   $TCPParametersKey).DeadGWDetectDefault
  $DHCPDomain=$(Get-ItemProperty
                   $TCPParametersKey).DHCPDomain
  $DHCPNameServer=$(Get-ItemProperty
                   $TCPParametersKey).DHCPNameServer

  $DisableTaskOffload=$(Get-ItemProperty
                   $TCPParametersKey).DisableTaskOffload
  $EnableSecurityFilters=$(Get-ItemProperty
                   $TCPParametersKey).EnableSecurityFilters
  $ForwardBroadcasts=$(Get-ItemProperty
                   $TCPParametersKey).ForwardBroadcasts
  $IPEnableRouting=$(Get-ItemProperty
                   $TCPParametersKey).IPEnableRouter
  $DontAddDefaultGatewayDefault=$(Get
                   ItemProperty
$TCPParametersKey).DontAddDefaultGatewayDefault
  $EnableICMPRedirect=$(Get-ItemProperty
                   $TCPParametersKey).EnableICMPRedirect
  $Hostname=$(Get-ItemProperty
                   $TCPParametersKey).Hostname
  $Domainname=$(Get-ItemProperty
                   $TCPParametersKey).Domain
  $DomainNameD=$(Get-ItemProperty
     $TCPParametersKey).UseDomainNameDevolution
$NetBTParamKey="HKLM:\System\CurrentControlSe
                   \Services\netbt\parameters"
  $DHCPNodetype=$(Get-ItemProperty
                   $NetBTParamKey).DHCPNodeType
  $LMhostsEnab=$(Get-ItemProperty
                   $NetBTParamKey).EnableLMHosts
#Tipo di nodo
   $DHCPTipoNodo=""
 #Tipo di nodo:
 #1 b-node
 #2 p-node
 #4 m-node
 #8 h-node
 Switch ($DHCPNodetype)
       {
       1   {$DHCPTipoNodo="B-NODE"}
       2   {$DHCPTipoNodo="P-NODE"}
       4   {$DHCPTipoNodo="M-NODE"}
       8   {$DHCPTipoNodo="H-NODE"}
       Else {$DHCPTipoNodo="Sconosciuto"}
       }

#Routing abilitato?

 $IPRouting=""
 If ($IPEnableRouting -eq 0)
    {$IPRouting="No"}
 Else
    {$IPRouting="Sì"}
#Informazioni generali.
  $OSVersion= $(Get-WMIObject
                   Win32_OperatingSystem).Version
   "Windows IP Configuration"
  "-----------------------------------------------------
                                        ------------"
  " Operating System Version: $OSVersion"
  ""
  "        Host Name . . . . . . . . . . . . : $Hostname"
  "        Primary DNS Suffix  . . . . . . . :
                                        $DomainName"
  "        Tipo Nodo . . . . . . . . . . . . :
                                        $DHCPTipoNodo"
  "        Routing IP abilitato. . . . . . . : $IPRouting"
  "        Use DNS Domain Name Devolution. . :
                          $([boolean]$DomainNameD)"
  "        LMHosts Enabled . . . . . . . . . : $([boolean]
                                        $LMHostsEnab)"
  "        DNS Suffix Search List. . . . . . :
                                        $DomainName"
  ""
```

```
" Altri flag (0=Disabled):"
"------------------------------------------------------------------"
"         DisableTaskOffload  . . . . . . . : $DisableTaskOffload"
"         EnableSecurityFilters . . . . . . : $EnableSecurityFilters"
"         ForwardBroadcasts . . . . . . . . : $ForwardBroadcasts"
"------------------------------------------------------------------"
#Informazioni x adapter.
  $ListaNetworkAdapter=Get-WMIObject Win32_NetworkAdapter
  $ListaNetworkAdapterConfig=Get-WMIObject Win32_NetworkAdapterConfiguration
  $NumeroAdapter=$ListaNetworkAdapter.Length
  For ($Cont=0; $Cont -lt $NumeroAdapter; $Cont++)
      {
      $NIC=$ListaNetworkAdapter[$Cont]
$AdapterConfig=$ListaNetworkAdapterConfig[$Cont]
      #Informazioni su adapter IP Enabled
        If ($AdapterConfig.IPEnabled)
           {
           If ($OSVersion -gt 5.0)
              {$Conn=$Nic.NetConnectionID}
           Else
              {$Conn=$NIC.Index}
      #WINS resolution attiva?
$WINSResolution=$AdapterConfig.DNSEnabledForWinsResolution
             If ($WINSResolution)
                {$WINSProxy="Sì"}
             Else
                {$WINSProxy="No"}
           "------------------------------------------------------------------"
           "Adapter: $Conn"
           "------------------------------------------------------------------"
           "Adapter type. . . . . . . . . . . : $($Nic.AdapterType)"
           "Description . . . . . . . . . . . : $($Nic.Description)"
           "Physical Address. . . . . . . . . : $($Nic.MACAddress)"
           "Autoconfiguration Enabled . . . . : $($Nic.AutoSense)"
           "  -----------------------------------"
           "            CONFIGURAZIONE           "
           "  -----------------------------------"
           "  Connection-specific DNS Suffix  . : $($AdapterConfig.DNSDomain)"
           "  DHCP Enabled. . . . . . . . . . . : $($AdapterConfig.DHCPEnabled)"
           "  IP Address. . . . . . . . . . . . : $($AdapterConfig.IPAddress)"
           "  Subnet Mask . . . . . . . . . . . : $($AdapterConfig.IPSubnet)"
           "  Default Gateway . . . . . . . . . : $($AdapterConfig.DefaultIPGateway)"
           "  DHCP Server . . . . . . . . . . . : $($AdapterConfig.DHCPServer)"
           "  DNS Servers . . . . . . . . . . . : $($AdapterConfig.DNSServerSearchOrder)"
           "  Proxy WINS abilitato. . . . . . . : $WINSProxy"
           "  Primary WINS Server . . . . . . . : $($AdapterConfig.WINSPrimaryServer)"
           "  Secondary WINS Server . . . . . . : $($AdapterConfig.WINSSecondaryServer)"
           "  Lease Obtained. . . . . . . . . . : $($AdapterConfig.DHCPLeaseObtained)"
           "  Lease Expires . . . . . . . . . . : $($AdapterConfig.DHCPLeaseExpires)"
           "------------------------------------------------------------------"
           } # Fine If

      }#Fine For
```

```
------------------------------------------------------------------------
 Operating System Version: 5.1.2600

        Host Name . . . . . . . . . . . . : PRESARIO
        Primary DNS Suffix  . . . . . . . :
        Tipo Nodo . . . . . . . . . . . . : H-NODE
        Routing IP abilitato. . . . . . . : No
        Use DNS Domain Name Devolution. . : True
        LMHosts Enabled . . . . . . . . . : True
        DNS Suffix Search List. . . . . . :

 Altri flag (0=Disabled):
------------------------------------------------------------------------
        DisableTaskOffload  . . . . . . . : 0
        EnableSecurityFilters . . . . . . : 0
        ForwardBroadcasts . . . . . . . . : 0
------------------------------------------------------------------------
------------------------------------------------------------------------
Adapter: Connessione alla rete locale (LAN)
------------------------------------------------------------------------
Adapter type. . . . . . . . . . . : Ethernet 802.3
Description . . . . . . . . . . . : National Semiconductor Corp. DP83815/816 10
/100 MacPhyter PCI Adapter
Physical Address. . . . . . . . . : 00:0D:9D:8B:A5:FD
Autoconfiguration Enabled . . . . :
  -----------------------------------
            CONFIGURAZIONE
  -----------------------------------
  Connection-specific DNS Suffix  . :
  DHCP Enabled. . . . . . . . . . . : True
  IP Address. . . . . . . . . . . . : 10.10.11.7
  Subnet Mask . . . . . . . . . . . : 255.255.255.0
  Default Gateway . . . . . . . . . : 10.10.11.254
  DHCP Server . . . . . . . . . . . : 10.8.2.102
  DNS Servers . . . . . . . . . . . : 10.8.2.101 10.8.2.102
  Proxy WINS abilitato. . . . . . . : No
  Primary WINS Server . . . . . . . : 10.8.55.4
  Secondary WINS Server . . . . . . : 10.8.55.12
  Lease Obtained. . . . . . . . . . : 20060612130830.000000+120
  Lease Expires . . . . . . . . . . : 20060703130830.000000+120
------------------------------------------------------------------------
```

Fig. 3: ***Output dello script programma Display_IP_Config.ps1.***

Adesso cerchiamo di andare avanti analizzando questo codice.

CONSIDERAZIONI SULLO SCRIPT

Partiamo dalla dichiarazione delle variabili. Come si sarà notato, non è difficile intuire che per dichiarare una qualsiasi variabile è necessario anteporre il carattere $ al nome della variabile stessa. Ovviamente il discorso non è così semplice ed occorrerebbe aggiungere

molto di più sull'argomento, ma per gli scopi di questa trattazione questo è quanto serve sapere.
Già dalla dichiarazione della prima variabile, la $TCPParametersKey, qualcuno potrebbe intuire qualcosa d'importante.
Apparentemente (e così è) contiene il riferimento ad una chiave del Registry secondo la sintassi HKLM:<Path To Key>. Quello che potrebbe essere meno chiaro è il suo utilizzo.
Con l'introduzione della PowerShell è possibile navigare all'interno del Registry (limitatamente alle chiavi HKEY_LOCAL MACHINE e HKEY_CURRENT_USER) come se si trattasse di un insieme di cartelle e sottocartelle.
Se dalla shell digitiamo il comando HKLM: e successivamente lanciamo una DIR, otterremo proprio l'elenco delle chiavi principali della HKEY_LOCAL_MACHINE. Stesso discorso per HKCU.
Tornando quindi al nostro script, abbiamo visto quindi che la $TCPParametersKey contiene proprio il riferimento alla chiave del Registry che memorizza alcune delle informazioni sulla configurazione di rete.
L'istruzione successiva utilizza questo dato in un modo che evidenzia la potenza della sintassi e dei comandi del nuovo linguaggio di scripting:

```
$DataBasePath=$(Get-ItemProperty
                    $TCPParametersKey).DataBasePath
```

Innanzitutto cominciamo dal cmdlet Get-ItemProperty. Questo command-let consente di recuperare informazioni su di un file o comunque su un oggetto permettendo di poterle trattare in seguito opportunamente. Per comprendere meglio il suo funzionamento ci avvarremo di un esempio che mostri realmente quello che accade.
Dalla PowerShell lanciamo il comando:

```
Get-ItemProperty
    "HKLM:\System\CurrentControlSet\Services\tcpip\
                                        paameters"
```

Il risultato sarà qualcosa del tipo:

```
PSPath                  :
 Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MA
                                            CHINE\
  System\CurrentControlSet\Services\tcpip\parameters
PSParentPath            :
 Microsoft.PowerShell.Core\Registry::HKEY_LOCAL_MA
                                            CHINE\
System\CurrentControlSet\Services\tcpip
PSChildName                         : parameters
PSDrive                             : HKLM
PSProvider                          :
                 Microsoft.PowerShell.Core\Registry
NV Hostname                         : PRESARIO
DataBasePath                        :
               C:\WINDOWS\System32\drivers\etc
NameServer                          :
ForwardBroadcasts                   : 0
IPEnableRouter                      : 0
Domain                              :
Hostname                            : PRESARIO
SearchList                          :
                             MyDomain,MyDomain.it
UseDomainNameDevolution             : 1
EnableICMPRedirect                  : 1
DeadGWDetectDefault                 : 1
DontAddDefaultGatewayDefault        : 0
EnableSecurityFilters               : 0
DisableTaskOffload                  : 0
PerformRouterDiscovery      : 0
DhcpNameServer                      : 10.0.0.101
                                      10.0.0.102
DhcpDomain  yDomain.it.
```

Arrivati sin qui, non dovrebbe essere molto difficile capirne la logica. Il Get-ItemProperty ritorna l'oggetto che rappresenta l'entry alla chiave del Registry come un'entità con delle proprietà (pari alle sottochiavi trovate) e le utilizza per memorizzarne le impostazioni. Apparentemente complicato, ma estremamente flessibile.
In questo modo lo script può recuperare dal registro di Windows tutte le impostazioni di cui necessita. Andando avanti s'incontra un costrutto che ricorda molto il linguaggio C: il costrutto SWITCH. Esso può essere visto più come una classica SELECT che come una variante del costrutto IF. Il suo funzionamento è semplice e non sprecheremo ulteriore spazio. Appena all'inizio della sezione Informazioni generali notiamo un'altra importante novità: il cmdlet Get-WMIObject.
Questo cmdlet è tra quelli che probabilmente incontreranno il consenso della maggior parte degli sviluppatori di script e batch perché permette di ridurre drasticamente il numero di linee di codice che sfrutta WMI.
Come fatto in precedenza, partiremo da un esempio:

```
Get-WMIObject Win32_OperatingSystem
```

produce a video qualcosa del tipo:

```
SystemDirectory   : C:\WINDOWS\system32
Organization              : Bull Italia Spa
BuildNumber       : 2600
```

```
RegisteredUser     : Francesco Lippo
SerialNumber       : 11111-OEM-2222222-33333
Version                              : 5.1.2600
```

Forti delle considerazioni precedenti, il comando:

```
(Get-WMIObject Win32_OperatingSystem).Version
```

non potrà che produrre come output la stringa 5.1.2600 (importantissimo l'uso delle parentesi per delimitare la classe a cui WMI dovrà far riferimento). Se qualcuno è pratico di programmazione di script e faceva uso tempo fa di WMI, ricorderà certamente che produrre lo stesso risultato con lo stesso numero di righe, appena visto, è praticamente impossibile.
Andando ancora avanti nell'esame dello script, osserviamo che recupera anche la versione del sistema operativo installata. Per inciso, prima di proseguire, aggiungiamo che, se avessimo voluto memorizzare solo le prime cifre del campo Version (prive cioè del BuildNumber), avremmo dovuto semplicemente considerare:

```
$(Get-WMIObject
       Win32_OperatingSystem).Version.Substring(0,3)
```

Adesso, spostiamo la nostra attenzione nella sezione Informazioni x adapter. Le due istruzioni seguenti

```
$ListaNetworkAdapter=Get-WMIObject
                              Win32_NetworkAdapter
$ListaNetworkAdapterConfig=Get-WMIObject
                    Win32_NetworkAdapterConfiguration
```

si preoccupano di recuperare l'elenco dei network adapter e della relativa configurazione di rete. Poiché l'elenco potrebbe essere piuttosto lungo, il ciclo FOR-NEXT successivo effettua il controllo soltanto sui device IP Enabled, scartando i restanti. Il funzionamento del ciclo dovrebbe essere abbastanza intuitivo e non verrà aggiunto altro in merito.
Prima di passare alle conclusioni, ancora qualche piccola considerazione. Sinora non abbiamo fatto cenno all'avvio dei propri script dalla PowerShell. Se ad esempio il nostro file si chiama DisplayIPConfig.ps1 e si trova nella stessa directory in cui siamo al momento, è sufficiente lanciare uno qualunque di questi comandi:

```
.\DisplayIPConfig
./DisplayIPConfig
```

L'estensione PS1 non è necessaria. Invece, nel caso il file si trovi in una directory diversa da quella corrente, (ad esempio F:\PowerShell Scripts) sarà sufficiente lanciare un comando simile a:

```
&"F:\PowerShellScripts\DisplayIPConfig"
```

Dedichiamo ancora qualche istante ad un cmdlet che potrebbe passare inosservato: Get-member.
Questo command-let consente di recuperare proprietà e metodi di qualunque oggetto in maniera molto semplice.
Vediamone subito un piccolissimo esempio:

```
Get-Service|Get-Member
```

produce qualcosa del tipo:

```
Name                 MemberType    Definition
----                 ----------         ----------
Name                                    AliasProperty
                               Name = ServiceName
add_Disposed                            Method
         System.Void add_Disposed(EventHandle...
Close                                   Method
                                System.Void Close()
Continue                                Method
                             System.Void Continue()
...
CanPauseAndContinue         Property
        System.Boolean CanPauseAndContinue {...
CanShutdown                 Property
              System.Boolean CanShutdown {get;}
CanStop                     Property
                   System.Boolean CanStop {get;}
Container                   Property
        System.ComponentModel.IContainer Con...
```

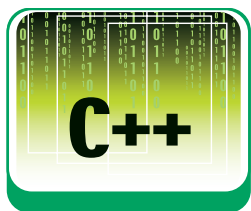
Inutile ribadire che queste informazioni possono tornarci utili nei nostri script e pertanto è davvero importante comprendere la validità di questo cmdlet.

CONCLUSIONI

Non potevamo non concludere questo articolo con un'affermazione più che ovvia: e non finisce qui. La PowerShell farà sicuramente parlar di sé per molto tempo e certamente non potranno che essere per la maggior parte critiche positive. Chiunque decida di avventurarsi in questo nuovo mondo scoprirà presto i vantaggi in termini di versatilità e potenzialità offerte dal nuovo linguaggio di scripting e non potrà più farne a meno.

Francesco Lippo

SINGLETON PATTERN COME USARLO?

NON MOLTI CONOSCONO QUESTA TECNICA DI PROGRAMMAZIONE. TUTTAVIA SI TRATTA DI UN TASSELLO FONDAMENTALE PER NON INCIAMPARE IN OGGETTI DUPLICATI E STRUTTURE POCO EFFICIENTI. IMPARIAMO COSA È E COME FUNZIONA

Nel precedente appuntamento abbiamo avuto modo di discutere cosa sia un pattern e quale sia la sua utilità, analizzando il pattern Observer. Questa volta descriveremo l'utilizzo di un altro tipo di pattern: il singleton. Ricordiamo brevemente che i pattern sono in sostanza soluzioni standard a delle problematiche frequenti in campo software. Essi si basano principalmente sulla programmazione orientata agli oggetti (OOP) e mirano a migliorare la qualità del codice agendo su caratteristiche legate ad essa. Ad esempio mirano ad aumentare la riusabilità del codice, ad aumentare il livello di incapsulamento delle informazioni e a diminuire il livello di coesione tra oggetti di classi differenti.
Il singleton è un modello di realizzazione di una classe che prevede la centralizzazione di un certo tipo di operazioni, attraverso l'utilizzo di un'istanza di un oggetto, che sia unica all'interno del programma, e per tutto il ciclo di vita del programma. In altre parole si utilizza un singolo oggetto per determinate operazioni, ma:

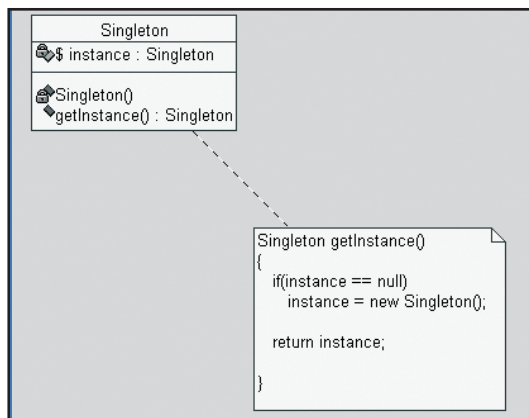
- non ci possono essere due oggetti di questo tipo
- questo oggetto si può creare una volta solamente
- gli altri oggetti accedono alle funzionalità del singleton, che quindi ha accesso unico alla risorsa che gestisce

A scapito della semplicità della struttura di un singleton, il cui diagramma UML è riportato in **FIGURA**, la sua utilità è davvero elevata. Vediamo ad esempio una problematica concreta, in cui applicare questo pattern porta a una soluzione di qualità elevata.

LA PROBLEMATICA

Supponiamo di volere implementare in una nostra applicazione una funzionalità di logging. Il "log" di un programma, come noto, è generalmente un file di testo contenente una sequenza di stringhe descrittive dell'attività svolta dal programma stesso. A ciascuna di queste stringhe è associato un "timestamp" ovvero una stringa contenente l'istante preciso nel quale l'evento che è descritto nel log si è verificato. Quello che segue, ad esempio, è un estratto del file di log del firewall software installato sul PC di chi scrive:

```
06/29/2006 20:24:49 Blocked ICMP Incoming
150.101.120.105 3 192.168.1.103
06/29/2006 20:24:49 Allowed ICMP Incoming
150.101.120.105 3 192.168.1.103
06/29/2006 20:24:49 Allowed UDP  Incoming
192.168.1.103 138 192.168.1.255
06/29/2006 20:24:49 Allowed UDP Outgoing
192.168.1.255 138 192.168.1.103
```

Si può notare la presenza di informazioni più o meno utili, come ad esempio se il pacchetto di rete è in ingresso o uscita (Incoming/Outgoing), se è stato o meno bloccato (Allowed/Blocked) ecc. All'inizio di ciascuna riga vi è il timestamp che permette di capire quando si è verificato un dato evento, al fine di ricostruire eventuali situazioni anomale o correggere errori nella configurazione del programma.
È evidente che la funzionalità di logging va al di là di una semplice stampa di una stringa.

Infatti si va incontro a diverse problematiche:

- deve essere aperto un file in scrittura
- a ogni stringa che si stampa deve essere associata una data e un orario
- la stampa deve essere fatta in maniera automica, cioè non ci devono essere sovrapposizioni di stringhe se due oggetti "loggano" contemporaneamente
- la stampa non deve inficiare le prestazioni dell'applicazione o causarle problemi
- ecc.

Date queste caratteristiche è ragionevole pensare di incapsulare le varie funzionalità di logging all'interno di una classe. L'istanza di questa classe, che verrà realizzata come un singleton, costituirà l'unico e solo punto di accesso di un qualsiasi oggetto della nostra applicazione alle funzionalità di logging.

LA SOLUZIONE IL SINGLETON

Una possibile classe che risolva il nostro problema è la seguente:

```
// log.h
#ifndef LOG_H
#define LOG_H

#include <list>
#include <string>

class Log {
public:
  void Scrivi(const char* linea);
  bool Salva(const char* nomefile);
private:
  std::list<std::string> m_data;
};
#endif

// log.cpp
#include <fstream>
using namespace std;

#include "log.h"

void Log::Scrivi(const char* linea);
{
m_data.push_back(linea);
}

bool Log::Salva(const char* nomefile);
{
  ofstream file(nomefile, ios::out);
  list<string>::iterator it;
  list<string>::iterator lastIt=m_data.end();
  for (it=m_data.begin(); it!=lastIt; ++it)
    file << *it << endl;
  file.close();
}
```

Questa classe fornisce a tutti gli effetti le varie funzionalità di scrittura di un log, anche se risulta un po' naif come implementazione (ad esempio manca il timestamp ed è necessario salvare "manualmente" il file di log chiamando la funzione Salva()).
Tuttavia il principale difetto di questa realizzazione è che essa accorpa le funzionalità richieste in un'unica classe ma NON in un'unica istanza di quella classe. In altre parole è possibile istanziare quanti oggetti di tipo Log si vuole, senza alcun vincolo. Questo è in contraddizione con quanto detto sinora sulla bontà del pattern Singleton, per cui ora modificheremo il codice di questa classe in maniera tale che essa risponda alle nostre esigenze.
Per fare in modo che un oggetto di tipo Log venga istanziato una e una sola volta è possibile utilizzare il meccanismo delle variabili statiche del C++. Ricordiamo brevemente che anteponendo, nella dichiarazione di una variabile, la parola chiave "static" si istruisce il compilatore a fare in modo che la variabile in questione sia dichiarata solo la prima volta che quel codice viene eseguito.
Potremmo quindi scrivere:

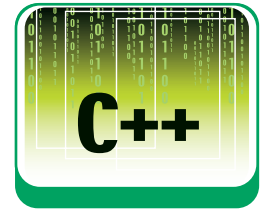
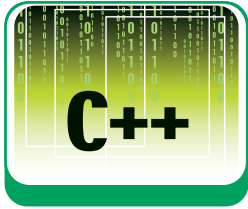
```
class Log {
public:
  static void Scrivi(const char* linea);
  static bool Salva(const char* nomefile);
private:
  static std::list<std::string> m_data;
};
```

E nel file Log.cpp aggiungere:

```
std::list<std::string> Log::m_data;
```

Abbiamo reso statiche le dichiarazioni di funzione e la variabile della classe Log che immagazzina le stringhe da stampare. Un metodo statico di una classe può essere utilizzato senza istanziare l'oggetto della classe stessa, utilizzando l'operatore "::" :

```
Log::Scrivi("Ciao!");
```

La variabile membro m_data viene invece istanziata direttamente nel file .cpp nel quale sono implementate le funzioni.
Questa soluzione da una parte soddisfa i

nostri requisiti, dall'altra però risulta poco flessibile sotto altri punti di vista. Ad esempio il fatto che il membro sia statico rende difficile controllarne l'inizializzazione e la distruzione, affidando quasi completamente al compilatore il compito di posizionare queste due fasi all'interno del ciclo di esecuzione dell'applicazione.
Un altro svantaggio risiede nel fatto che utilizzare metodi statici impedisce di utilizzare una delle caratteristiche più interessanti della programmazione a oggetti, e cioè il polimorfismo, attraverso l'uso di funzioni "virtuali".

SINGLETON POLIMORFICO

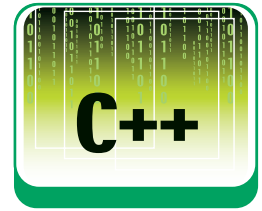
Le funzioni virtuali del C++ (cioè le funzioni di una classe dichiarate con la parola chiave "virtual") servono ad attuare il meccanismo del "late binding". Questo meccanismo consente di decidere a run-time, cioè durante l'esecuzione del programma, quale sia il metodo corretto da richiamare. Questo è l'opposto dell' "early binding" in cui la decisione viene fatta al momento della compilazione del codice.
Supponiamo ad esempio di volere realizzare due differenti tipi di log, uno che viene salvato su file e l'altro che invece viene inviato via rete a una macchina remota, attraverso una connessione TCP. Le funzioni Scrivi() e Salva() saranno senz'altro comuni ai due tipi di log, mentre il log che effettua la connessione avrà bisogno di ulteriori informazioni per potere essere utilizzato, ad esempio l'indirizzo IP della macchina remota. Per realizzare questo meccanismo possiamo astrarre le funzionalità comuni ai due tipi di log, inserendoli in una classe base astratta. Successivamente creiamo due classi derivate, che implementano le funzionalità comuni e aggiungono, all'occorrenza, le ulteriori funzionalità richieste:

```
// Log.h
class Log {
public:
  virtual ~Log () {}
  virtual void Scrivi(const char* linea)=0;
  virtual bool Salva(const char* nomefile)=0;
};

// LogFile.h
class LogFile : public Log {
public:
  virtual ~LogFile();
  virtual void Scrivi(const char* linea);
  virtual bool Salva(const char* nomefile);
private:
  std::list<std::string> m_data;
};

// LogNetwork.h
class LogNetwork : public Log {
public:
  virtual ~LogNetwork();
  virtual void Scrivi(const char* linea);
  virtual bool Salva(const char* nomefile);
  virtual void ImpostaPCRemoto(const
                              TCPConnection& server);
private:
  TCPConnection m_logServer;
  std::list<std::string> m_data;
};
```

In questo esempio la classe base astratta Log contiene le due funzioni virtuali pure (cioè non implementate) Salva() e Scrivi(). LogFile e LogNetwork sono le classi derivate che implementano il codice di queste funzioni, ciascuna con le sue peculiarità. Ad esempio LogFile farà un accesso a disco per salvare il contenuto del log, mentre LogNetwork invierà la richiesta di salvataggio tramite rete al PC remoto. Da notare come il distruttore della classe base debba essere dichiarato anch'esso come "virtual", per permettere all'oggetto istanziato di utilizzare il proprio distruttore anziché quello della classe base.
In definitiva, riprendendo il discorso precedente, possiamo beneficiare di un meccanismo importante come quello del polimorfismo anche utilizzando il pattern singleton. Per fare questo tuttavia dobbiamo rinunciare all'utilizzo di metodi e membri statici.
Vediamo quindi come aggirare questo problema.

IL CODICE

Il trucco per evitare di rendere tutte le funzioni del singleton statiche, consiste nell'utilizzare il meccanismo statico solamente in fase di creazione del singleton. La creazione non avviene tramite un normale costruttore, ma tramite una particolare funzione, chiamata Istanza(), che restituisce un puntatore valido all'oggetto singleton. Istanza() deve essere di tipo statico, in quanto deve essere chiamata prima della creazione del singleton, creazione della quale si occupa essa stessa. Applicando questi concetti alla nostra classe di esempio otteniamo:

```
class Log {

public:
```

```
  static Log* Instanza() {
    if (!m_pInstance)
      m_pInstance = new Log();
    return m_pInstance;
  }
  void Scrivi(const char* linea);
  bool Salva(const char* nomefile);
private:
  Log();
  Log(Log const&);

  static Log* m_pInstance;
  static std::list<std::string> m_data;
};

// in log.cpp
Log* Log::m_pInstance = NULL;
```

La funzione Istanza() controlla se il puntatore all'oggetto Log m_pInstance è NULL oppure no. Se è NULL, cioè se è la prima volta che Istanza() viene chiamata, l'oggetto viene creato e ne viene restituito il riferimento. In caso contrario viene restituito il riferimento memorizzato in precedenza. Un esempio di utilizzo di questa classe è il seguente:

```
Log::Istanza()->Scrivi("Ciao!");
Log::Istanza()->Salva("log.txt");
```

La prima chiamata a Istanza() provocherà la creazione dell'oggetto Log, sul quale verrà richiamata la Scrivi(). La seconda chiamata a Istanza() utilizzerà il riferimento creato in precedenza per richiamare la Salva().
Una cosa fondamentale da notare è che sia il costruttore che il costruttore di copia sono dichiarati nella parte privata della classe. Questo impedisce di ottenere un riferimento all'oggetto se non attraverso la funzione Istanza(). In altre parole non è possibile scrivere:

```
Log* mio_log = new Log(); //NO!
```

poiché il costruttore è accessibile solamente al codice della classe Log stessa, ma non all'esterno. Un'altra cosa che non è possibile fare è effettuare una copia diretta di un oggetto di tipo Log:

```
Log mio_log = *(Log::Istanza()); //NO!
```

Se ciò fosse possibile si avrebbe un metodo per costruire quanti oggetti vogliamo senza passare attraverso Istanza(), creando in questo modo più di un oggetto Log, cosa che è incompatibile col concetto stesso di Singleton.
Un appunto che può essere mosso alla particolare realizzazione del pattern singleton che abbiamo esposto sinora è che esso può creare dei "memory leak", cioè "sprechi di memoria". Non essendo possibile controllare direttamente la distruzione del singleton, questo rimarrà in memoria anche quando non sarà utilizzato. Questo problema tuttavia concerne molto più la definizione di "memory leak" che la realizzazione del pattern. Se infatti consideriamo un memory leak come la presenza di memoria non deallocata direttamente dal software, allora abbiamo, nel nostro caso, un memory leak. Se invece consideriamo un memory leak come l'esistenza di una generica memoria non più controllabile, allora NON siamo in presenza di un memory leak, poiché il nostro singleton "vive" per tutta la durata dell'esecuzione del software. La memoria viene "sprecata" unicamente nel momento in cui l'esecuzione termina. Tuttavia nella maggior parte dei sistemi operativi TUTTA la memoria allocata per un processo viene liberata nell'istante stesso in cui quel processo termina, e viene resa disponibile per altre operazioni. Quindi tecnicamente lo spreco di memoria non avviene mai.

CONCLUSIONI

Il pattern singleton è molto utile in tutti quei casi in cui si ha bisogno di centralizzare una serie di funzionalità all'interno di una architettura software. Questo può avvenire ad esempio quando si ha a che fare con una risorsa unica, di tipo sia logico, ad esempio il file di log dei nostri esempi, che fisico come ad esempio la memoria della scheda video o una particolare periferica. Abbiamo visto che in C++ la realizzazione del pattern singleton passa attraverso due concetti base:

- l'utilizzo di una funzione statica Istanza() che si occupa di restituire un riferimento valido al singleton
- la dichiarazione di costruttore e costruttore di copia come metodi privati della classe cui si riferiscono

I singleton sono spesso utilizzati anche nella realizzazione di librerie o kit di sviluppo software, per cui è bene conoscerne la natura, sia per utilizzarli al meglio sia per progettare noi stessi delle librerie di qualità. Inoltre lo incontrerete spesso nell'utilizzo di particolari librerie o framework esterni poiché si tratta di uno dei pattern più usati in programmazione

Alfredo Marroccelli

SOFTWARE SUL CD

Flex 2.0

IL NUOVO FRAMEWORK DI ADOBE

Straordinario! Non ci sono parole per descrivere questo nuovo prodotto della linea Adobe che in un sol colpo fa piazza pulita di un vecchio modo di pensare alle applicazioni per inventarne uno tutto nuovo, semplice ed estremamente potente. Chi ha letto questo numero di ioProgrammo sa già come funziona Flex 2.0. Si scrive un programma utilizzando un dialetto di XML, lo si da in pasto al compilatore e si ottiene in uscita un file SWF ovvero utilizzabile dal noto player Flash di macromedia. Quali sono i vantaggi? Ce ne sono diversi, i più facili da intuire sono la completa indipendenza dal sistema operativo senza per questo doversi affidare alla macchinosità di java. La possibilità che la stessa applicazione possa girare praticamente senza modifiche sia in modo standalone che nel browser. La possibilità di sviluppare interfacce complesse come quelle che siamo abituati a vedere in flash e tali che conservino lo stesso aspetto e comportamento su praticamente ogni sistema operativo. Dal punto di vista esclusivamente funzionale rispetto alla classica programmazione Flash le innovazioni sono importanti. L'interazione con i database è di una facilità estrema, c'è una completa indipendenza del codice dal layout, i programmatori hanno a disposizione una gamma di strumenti veramente ampia per poter svolgere il loro lavoro. Si tratta dunque di un prodotto decisamente innovativo, assolutamente da provare. In questo numero di ioProgrammo pubblichiamo un articolo che vi metterà subito in grado di utilizzare il prodotto e di intuirne le straordinarie possibilità
Directory:/FLex

AJAX PRO STARTER KIT

PER INIZIARE FACILMENTE CON VISUAL STUDIO

Da qualche tempo non si fa altro che parlare di Ajax. Si tratta della nuova tecnologia che consente di realizzare pagine web dinamiche che non necessitano del reload per aggiornare i dati. Come potete intuire si tratta di un'innovazione senza precedenti che consente di realizzare applicazioni Web con comportamenti molti simili alle normali applicazioni Desktop. Visual Studio 2005 dispone della tecnologia degli Starter Kit, che consentono di iniziare un progetto partendo da una sorta di template che ne facilità l'avvio. Il tool che vi presentiamo è appunto lo starter kit che consente di iniziare a sviluppare un progetto Web in ASP.NET ed in tecnologia AJAX. Si tratta senza dubbio di un valido aiuto per coloro che vogliono sviluppare le proprie Web Application
Directory:/AjaxProStarterKit

ASCEND.NET WINDOWS FORM

UN'UTILE COLLEZIONE DI CONTROLLI

Si tratta di una serie di controlli da integrare in Visual Studio.NET come non se ne vedeva da tempo. All'interno di questa libreria trovano spazio 6 nuovi componenti. In particolare: un gradient Panel, un gradientCaption, un gradient SplitBar, un gradientNavigationButton, un gradientAnimation Pane image e infine un Navigation Pane. Tutti componenti tesi a migliorare l'aspetto estetico di un'applicazione aggiungendovi quella morbidezza che favorisce un miglior rapporto fra l'utente ed il software
Directory:/AscendNet

COMMERCE STARTER KIT 1.0.0.3

IL TEMPLATE PER PAYPAL

Con il nome in codice CSK, questo starter kit contiene un template per lo sviluppo di un sito di commercio elettronico con interfaccia verso paypal. Scritto per la versione 2.0 di Asp.NET si configura come un prodotto completo dotato di tutti gli elementi classici di un negozio di commercio elettronico, dal più classico dei carrelli fino alla gestione del catalogo e degli ordini. Rimane ovviamente uno starter kit, quindi è compito del programmatore personalizzare il comportamento dei singoli task. E questo per noi sviluppatori è ulteriore motivo di interesse, non si tratta infatti solo di un prodotto pronto all'uso ma di un punto di partenza in grado di essere adattato ala proprie esigenze
Directory:/commercestarterkit

ATLAS CONTROL TOOLKIT

I MIGLIORI COMPONENTI PER IL WEB

Nasce da una collaborazione fra Microsoft e i creatori originali, questo interessante pacchetto che si propone di fornire al programmatore una serie di

componenti "client" adatti a favorire lo sviluppo di Web Application. All'interno del pacchetto si trova una ricca collezione di esempi che fanno uso di una quindicina di nuovi controlli, infine c'è un SDK che consente di estendere ulteriormente Atlas che in questo modo si configura come un vero e proprio Framework per lo sviluppo

ECLIPSE 3.2

ADESSO PUOI SVILUPPARE ANCHE IN FLASH

In moltissimi già conoscono Eclipse con l'IDE più utilizzato dai programmatori Java, tanti altri lo hanno esteso per le proprie esigenze. In molti lo utilizzano per PHP o per C++, e chi ha letto l'articolo su "Flex 2.0" presente in questo stesso numero di ioProgrammo, saprà adesso che è possibile utilizzare Eclipse anche per sviluppare applicazioni mxml cioè in un linguaggio che una volta compilato restituisce dei normai file SWF.
E con questo la maturità raggiunta da Eclipse diventa ormai completa, tanto che non esiste linguaggio, problematica o applicazione che non possa essere in una qualche maniera, senza dubbio ottimale, affrontata con Eclipse. Davvero un bel salto in avanti per questo progetto, che nel tempo è diventato un punto di riferimento per un'intera comunità

Directory://Eclipse

DEVCPP 4.9.2

IL PIÙ AMATO DAI PROGRAMMATORI C++

Da qualche tempo non faceva capolino fra le nostre pagine questo interessantissimo IDE per programmatori C++. Sicuramente si tratta di uno degli ambienti più amati dai programmatori di questo linguaggio. Dalla sua parte un'incredibile leggerezza che lo rende utilizzabile su macchine anche non eccezionalmente performanti, e poi tutta una serie di features che vanno dal code complexion alla sintax highlightinhg, ma non solo. Si tratta di un software completo che esteso con plugin diventa anche un editor RAD con interfaccia visuale. ioProgrammo spesso lo usa come editor di riferimento

Directory://devcpp

ECLIPSE MODELING FRAMEWORK

IL TOOL PER PRODURRE CLASSI A PARTIRE DA UN MODELLO

Eclipse fa riferimento a questo framework quasi per ogni operazione che ha necessità di svolgere. Si tratta si un tool che consente di generare delle classi partendo da un modello specificato in formato XMI. I modelli possono essere generati utilizzando un qualunque tool e qualunque applicazione supporti il formato EMF sarà in grado di condividere le stesse informazioni. Si tratta dunque di un progetto che ha una parte importante in Eclipse ma che può essere utilizzato in modo conveniente per la risoluzione di tutta una serie di problemi

Directory:/Eclipse

GRAPHICAL EDITING FRAMEWORK

IL PLUGIN PER FAR DIVENTARE ECLIPSE RAD

Non si tratta di un RAD completo ma di un framework a cui Eclipse può appoggiarsi per creare ambienti RAD per i linguaggi supportati. Supponete per esempio di voler costruire un plugin di Eclipse che consenta di creare un ambiente RAD per lo sviluppo in PYTHON. Dove per RAD si intende un ambiente dotato di editor visuale che vi metta in grado di creare interfacce trascinando gli elementi su una Form. Per far questo avete bisogno di GEF che vi mette a disposizione tutti gli strumenti per trasformare il vostro IDE preferito in un RAD leggendo la descrizione del modello a partire da un file XML. GEF ad esempio è indispensabile per installare FlexBuilder, l'IDE RAD per flex

Directory:/gef

HIBERNATE 3.2.0

TRASFORMA I DA DATI DA SQL AD OGGETTI

Ormai lo sappiamo abbiamo letto tutti gli articoli di ioProgrammo su Hibernate, pensare in termini di query e dati SQL non è più conveniente! Disponiamo di oggetti e di classi, non possiamo continuare a lavorare in modo disomogeneo trattando i dati SQL in maniera separata dal resto del flusso del programma. E per mappare i dati da SQL ad oggetti abbiamo bisogno di un tool. Il primo ed ancora adesso Leader di questo settore è Hibernate, attraverso il quale riusciamo ad ottenere comodamente i risultati voluti

Directory:/ Hibernate

OPENLASZLO 3.3.1

L'ALTERNATIVA OPENSOURCE A FLEX

A Flex dedichiamo veramente tantissimo spazio in questo numero di ioProgrammo. Sappiamo già che è un compilatore grazie al quale è possibile ottenere applicazioni Flash partendo da un file descrittivo in XML. Sappiamo anche che il Flex SDK è gratuito, allo stesso modo FlexBuilder ovvero l'editor per Flex ha un costo! Non è un grande problema, in quanto l'uso dell'editor non è vincolante per poter scrivere i programmi, tuttavia è comunque un vincolo. Ed ecco arrivare Ide4laszlo, alternativa OpenSource a Flex che ha dalla sua parte oltre la gratuità degli strumenti, anche la possibilità di ottenere interfacce DHTML anziché Flash.
Si tratta di un buon prodotto a cui sicuramente dedicheremo spazio nelle nostre pagine

Directory:/ ide4laszlo

PHP 5.1.4

IL LINGUAGGIO DI INTERNET

Se seguite ioProgrammo o più semplicemente siete dei programmatori Web, o ancora molto più semplicemente navigate su Internet, non potete non sapere che cosa è PHP. Si tratta del linguaggio con il quale sono sviluppate la maggior parte delle applicazioni internet esistenti. Quasi tutto il software per il web si regge su PHP. La curva di apprendimento è bassissima, le funzionalità esposte elevatissime, certamente se avete intenzione di sviluppare per il web non potrete fare a meno di provare anche questo linguaggio come base per le vostre applicazioni. Attualmente la versione 5.1.4 espone un modello ad oggetti piuttosto completo che rende PHP un linguaggio moderno dalla curva di apprendimento molto rapida e persino didattico per chi vuole imparare a programmare ad oggetti senza dover affrontare la complessità di Java o .NET. Ed è in arrivo lo Zend Framework...

Directory PHP 5.1.4

CRITTOGRAFIA ASIMMETRICA

DALL'ALGORITMO DI MERKLE HELLMAN, ALLA CRITTOGRAFIA RSA , PASSANDO IL METODO DEL PUZZLE E DI BLUM GOLDWASSER, ESPLORIAMO GLI ALGORITMI DI CRITTOGRAFIA ASIMMETRICA CHE CONSENTONO LO SCAMBIO SICURO DI DATI, O CHE TENTANO DI FARLO

REQUISITI

Conoscenze richieste
Basi di crittografia

Software
-

Impegno

Tempo di realizzazione

Il modo di affrontare un problema ne individua la tecnica. Tra queste pagine abbiamo esplorato molte delle tecniche utilizzate nel mondo della programmazione. ÈMolte trattazioni degli scorsi appuntamenti convergono verso l'affascinante mondo dei crittosistemi. La ricerca di numeri primi e il problema della fattorizzazione così come l'analisi del mese scorso circa gli algoritmi greedy ci suggeriscono di approfondire alcuni importanti metodi per la trasmissione sicura dei dati. Vedremo come una formulazione leggermente differente del problema di knapsack (il problema del ladro) ci introdurrà una pietra miliare della crittografia a chiave pubblica, ovvero l'algoritmo di Merkle Hellman. Esploreremo altri metodi che afferiscono a tale ambito, alcuni poco conosciuti perché poco usati, ma molto interessanti da un punto di vista procedurale. E altri, già osservati in passato, come RSA più conosciuti e dalle innumerevoli applicazioni pratiche. La discussione non si esaurirà con questo articolo. Preannuncio per il prossimo numero l'interessante crittografia quantistica, che sta alimentando in questo periodo, un'intensa discussione.

MERKLE-HELLMAN

Si tratta di uno dei primi crittosistemi. È esplicativo circa le problematiche e le tecniche per la cifratura, anche se non ha prestazioni adeguate che ne possano giustificare un effettivo uso nel campo della sicurezza. Si tratta di un caso particolare del problema di Knapsack. È basato sulla somma di sottoinsiemi. Dato una lista di numeri è un numero z, questi sarà la somma di un sottoinsieme della lista. In altri termini z si ottiene come la somma di alcuni dei numeri che compongono la lista iniziale. Il problema è conosciuto come NP-completo. Si distinguono casi "facili" che possono essere risolti rapidamente. Vi sono però anche casi difficili. Si impone così lo schema di Merkle che tenta di trasformare casi difficili in facili e viceversa. Passiamo alle peculiarità del metodo. Inizialmente si considera un insieme crescente di numeri tranne lo zero, tale che ogni numero sia maggiore della somma dei precedenti.

```
x=(x1, x2, x3, … xn)
```

Un insieme (sacco) così composto si intende riempito con crescita esponenziale o (super crescita). Tale sequenza ricorda solamente quella di Fibonacci, per la quale ogni elemento è la somma dei due precedenti. Un esempio di tale sequenza è {1, 2, 4, 8, 16}. Si scelgono: un numero q casuale che sia maggiore o uguale della somma dell'insieme dei numeri appena indicata, è un numero intero y che sia coprimo con q, quindi MCD(y,q)==1.
La scelta di q assicura l'unicità del messaggio criptato. y avrà un inverso per permette l'operazione di decrittazione. La chiave pubblica si calcola come una sequenza:

```
b=(b1, b2, b3, …. bn)
```

dove il generico bi = y*xi (mod q). Quindi b è la chiave pubblica mentre l'insieme (y, x e q) è la chiave privata. Vediamo come avvengono le due operazioni di criptazione e decriptazione.

Sia a un messaggio:

```
a= (a1, a2, a3, … an)
```

con ai l'iesimo bit del messaggio. Ovviamente, ai può valere o zero o uno. Il messaggio criptato sarà la sommatoria di n elementi dei prodotti tra ai e bi.

```
c=Si(ai*bi)
```

QUANDO UN NUMERO È COPRIMO!

Due numeri sono coprimi tra di loro se non hanno alcun divisore comune. O in altri termini se il massimo comun divisore tra i due numeri è uno. La notazione per esprimere questa proprietà tra due numeri a e b è quindi: MCD(a,b)==1

In altri termini c è la somma dei valori di b in corrispondenza degli indici i per i quali il messaggio ha bit pari a 1. In tal senso, e per la scelta di q, si comprende l'analogia con il problema del ladro come sommatoria di valori. La decrittografia si attua applicando la relazione inversa. Si calcola il valore s indicato come chiave privata.

```
s=y^-1 (mod q)
```

La soluzione si può affrontare in modo difficoltoso, si parla di complessità NP-hard se l'estrazione di a da c si ottiene attraverso uno knapsack riempito in modo casuale, mentre di complessità lineare se, come in questo caso, il sacco è riempito con la caratteristica di crescita esponenziale. Bisogna calcolare ad ogni passo della decriptazione c*s (mod q) e x=b*s (mod q). Ricordo che c è il cipher text e a il plain text. Con il sacco così come lo abbiamo riempito, ossia con la super crescita è facile (da un punto di vista algebrico) pervenire alla soluzione. Si considera il valore più grande di x, xk. Se xk è maggiore di c allora vorrà dire che il corrispondente ak=0 altrimenti se xk è minore di c allora ak=1. Fine primo passo. A questo punto si scorpora (sottrae) da c il prodotto ak*xk e si ripete l'operazione per l'elemento precedente di indice k-1. Si ripete l'operazione, decrementando k, fin quando non si è composto completamente a. Quando q è molto grande risulta difficoltoso calcolare s. La moltiplicazione modulare è il collo di bottiglia che può richiedere molto tempo.

CIRCA LA CRITTOGRAFIA A CHIAVE PUBBLICA

Un breve cenno per chi legge per la prima volta di crittografia. Sarà noto che si tratta dei metodi che vengono usati nel campo della sicurezza informatica. Ogni qual volta vogliamo inviare dati che desideriamo mantenere riservati, o semplicemente quando si intende archiviare dei dati in modo che eventuali occhi indiscreti non possano comprenderli si fa uso di metodi di crittografia. A dire il vero esiste anche un'altra grande famiglia di metodi che afferisce alla steganografia che ha il compito di "nascondere" le informazioni all'interno di altre informazioni di uso comune, come ad esempio la foto digitalizzata della nostra prima comunione. Ma questo è un capitolo che abbiamo già affrontato e che ci proponiamo in futuro di riaprire (ioProgrammo n. 71 - Fabio Grimaldi -). La crittografia semplicemente prende un testo in chiaro conosciuto come plain text, applica un algoritmo di crittografia con una chiave k e produce un testo cifrato o cipher text. Il testo può essere così inviato al destinatario o semplicemente può essere archiviato. A destinazione per ottenere il testo in chiaro sarà necessario applicare un algoritmo "inverso" di decrittografia che genera nuovamente il plain text. Nei primi metodi introdotti la chiave di cifratura e decifratura era identica. Per capire il concetto di chiave è sufficiente introdurre il più semplice degli algoritmi di crittografia, quello per sostituzione. Si supponga di avere il testo in chiaro "attuare il piano b" . Al fine di cifrare il testo si può pensare di sostituire ogni lettera del testo in chiaro con la lettera di due successiva secondo l'alfabeto inglese di uso comune. Il numero due non è altro che il valore della chiave (k=2). Il cipher text così ottenuto è: "cvvwct kn rkcpq d". Il metodo esposto ha un valore meramente didattico poiché è molto semplice da demolire, ad ogni modi ci fa capire il significato di chiave. Quando la chiave è identica si parla di metodo a chiave segreta. DES (Data encryption standard) sviluppato da IBM è il più conosciuto tra questi. Il problema principale per tali metodi è la necessità di individuare un canale sicuro per lo scambio tra mittente e destinatario delle chiavi. È evidente che non si può trattare dello stesso canale dove usualmente avvengono le comunicazione altrimenti non si capirebbe l'esigenza di cifrare i dati se questo venisse considerato sicuro. L'alternativa sono una serie di algoritmi per i quali non è richiesta la presenza di un canale sicuro me è sufficiente usare il canale standard che verrà usato anche per il setup del metodo. Appartengono a questa famiglia gli algoritmi a chiave pubblica come quello di Merkle-Hellman visto prima. Un esempio che introduce "magnificamente" l'ambito su cui ci muoviamo prevede lo scambio di dati attraverso posta standard tra due individui, Alice e Bob, coppia ormai conosciuta dai crittografi visto che è di fatto citata in letteratura ogni qual volta si devono fa-

ALGORITMO DEL PUZZLE

L' algoritmo del puzzle è un algoritmo di crittografia a chiave pubblica istruttivo poiché contiene molte delle idea alla base di tutti gli algoritmi asimmetrici ma che è difficilmente realizzabile. Fu Merkle a introdurlo nel 1974. La supposizione iniziale è sempre identica: Bob e Alice vogliano comunicare in modo sicuro, al riparo di Eva che avendo accesso al canale di comunicazione potrebbe intercettare scambi di informazione. Bob e Alice devono scambiarsi la chiave, è non avendo un canale sicuro devono usare quello a disposizione. Bob crea dei puzzle, ovvero tanti messaggi codificati con un algoritmo facile da forzare. Per puzzle nell'ambito proposto da Merkle si intende un messaggio cifrato abbastanza facile da risolvere come abbastanza facile forzare. Alice riceve questi puzzle, in numero molto elevato dell'ordine del milione, e ne sceglie uno a caso. Lo risolve, ovvero forza la cifratura. A questo punto Alice legge il messaggio contenuto nel puzzle che dice qualcosa del genere: "Sono il puzzle numero x, la chiave numero x è y". A questo punto Alice comunica a Bob il numero del puzzle: x. In questo modo Alice e Bob hanno una chiave in comune: y. Anche può conoscere la chiave y, avendo intercettato lo scambio dei puzzle, ma è mischiata tra un milione di altre chiavi. A questo punto potrebbe provare a risolvere tutti i puzzle fino a trovare il puzzle numero x, ma il numero dei puzzle è scelto in modo che questa operazione sia computazionalmente troppo lunga.

re esempi, e noi non siamo da meno! Come prima azione Alice mette nella cassetta il messaggio, la chiude con il suo lucchetto mediante chiave KA e la spedisce a Bob. Questi riceverà la cassetta applicherà il suo lucchetto con chiave KB e rispedirà la cassetta con i due lucchetti nuovamente ad Alice. Alice toglierà il suo lucchetto (visto che è in possesso della chiave B) e spedirà per l'ultima volta la cassetta a Bob, il quale non dovrà fare altro che aprire la cassetta con la sua chiave KB. Questo è solo un protocollo per sfruttare il canale pubblico, gli algoritmi di crittografica, nella nostra metafora sono i lucchetti. Ovviamente, la rivoluzione sta nell'aver comunicato segretamente avendo usato un canale pubblico, quindi per definizione insicuro. Un modo analogo per specificare lo stesso concetto è il seguente esempio: i lucchetti rappresentano le chiavi pubbliche e le chiavi le chiavi private (scusate la cacofonia). Alice chiede a Bob di spedirle il suo lucchetto, già aperto. La chiave dello stesso verrà custodita da Bob. Alice riceve il lucchetto e, con esso, chiude il la cassetta con il messaggio e la spedisce a Bob. Questi riceve il pacco e può aprirlo con la chiave di cui è unico proprietario.

L'ALGORITMO RSA

Il più conosciuto dei metodi di crittografia a chiave pubblica è opera di R. Rivest, A Shamir e L. Adleman, le cui iniziali hanno dato nome al metodo. Il tutto si basa su una funzione matematica la cui espressione algebrica è molto semplice. Come vedremo risulterà molto complessa la trattazione numerica. In associazione ad una funzione simile per la decrittazione, definisce i due algoritmi distinti di crittografia e di decrittografia. Siano P, C, e, d e n dei numeri e valga la seguente relazione:

$$C = P^e \bmod n \quad (1)$$

Che verrà adottata, per la crittazione.
Allora esisterà un numero d tale che:

$$P = C^d \bmod n \quad (2)$$

Funzione di decrittazione. Più avanti capiremo il significato delle diverse variabili. Sfruttando la proprietà di simmetria della funzione modulo si nota che le due funzioni di crittazione e decrittazione sono l'una l'inversa dell'altra e vale la proprietà commutativa.

$$P = C^d \bmod n = C = (P^e)^d \bmod n = (P^d)^e \bmod n$$

A questo punto si supponga che P sia il plaintext o blocchi appartenenti ad esso di lunghezza massima n. C sia il ciphertext o blocchi di esso; e e d siano delle variabili che come vedremo dovranno essere molto grandi. L'algoritmo RSA funziona come esposto nei passi che illustriamo di seguito:

1. Si considerano due numeri primi qualsiasi *p e q molto grandi dell'ordine di 10100.*
2. Si determina il prodotto $n=pq$.
3. Si determina il prodotto $m=(p-1)(q-1)$
4. Si calcola un numero *d* che sia minore di *m* ma che sia rispetto a *m* un numero primo. *d* ed *m* non hanno fattori primi in comune. d è dispari mentre *m* è pari.
5. Una volta scelto *d*, si determina un numero e che soddisfi l'equazione:
   $ed = 1 \pmod m$
   Ovvero e dovrà essere l'inverso di *d* modulo *m*.

La soluzione dell'equazione per determinare *e*, come detto prima, non è affatto semplice da un punto di vista numerico, soprattutto non lo è nel campo degli interi modulo *z*, dove *z* può assumere un qualsivoglia valore intero. Esistono comunque dei teoremi dell'algebra che ne danno una giustificazione matematica. Quindi, per cifrare si usa la relazione 1 da cui si ottiene il ciphertext *C* e per decifrare la 2 da cui si ottiene il plaintext *P*. Per cifrare bisogna conoscere la coppia di numeri e, n, mentre per fare l'operazione inversa devono essere noti *d*, *n*. Considerato che *n* è nota, l'idea rivoluzionaria sta nel rendere pubblica, quindi nota, anche *e*, ossia la chiave per cifrare, mentre rimarrà segreta *d*. La difficoltà è insita nell'operazione di scomposizione in fattori di *n* che è un numero grandissimo. Se si riuscisse ad analizzare la fattorizzazione di n si potrebbero individuare i due numeri *p* e *q* e conseguentemente *m* e quindi essendo noto e anche *d*; ma come detto tale operazione è talmente complessa da poterla definire "oggi" quasi impossibile. I calcoli di Rivest dicono che la fattorizzazione di un numero con 500 cifre richiederebbe 10^25 anni!! In definitiva vi sono due chiavi, una del trasmettitore che deve essere privata e che nel caso specifico è *d*, mentre è pubblica la chiave *e* (in alcuni casi si considera pubblica la coppia *e,n*) appartenente al destinatario. In definitiva Alice rende pubbliche le sue chiavi, la coppia (*e,n*) cosicché se qualcuno vuole inviarle un messaggio lo cifrerà mediante la relazione 1, supponiamo che a farlo sia Bob. Una volta che Alice riceverà il messaggio, con la sua coppia di chiavi private otterrà il plain text. A scopo puramente didattico facciamo un esempio sull'uso del metodo RSA. Per fare una prova, per semplicità di calcolo non considereremo chiavi che rispetteranno le specifiche del metodo, ovvero che siano dell'ordine di 10^100, ma saranno numeri molto piccoli, per poter fare i conti con maggiore agio. Ad ogni modo vedremo come anche così si presentano problemi. Ho fatto qualche prova che mi consentisse di individuare numeri "giusti". Ho preso p=5 e q=17che sono due numeri primi.
n=pq=85 mentre m=(p-1)(q-1)=64
Se si pone d=13 (questa è la chiave segreta non ditela

in giro) si ottiene dalla relazione ed=1 (mod 64) un valore di e pari a 5, provare per credere! Dovremo adottare un alfabeto di cardinalità n=45. Con i numeri scelti *P* non verrà associato a blocchi di caratteri ma ad un solo carattere. Supponiamo che la *a* sia codificata con il numero 1 la *b* con il numero 2 e così via e che il messaggio da cifrare sia "cia" (in tema di sicurezza mi sembra una scelta azzeccata!). La crittazione si otterrà applicando la relazione 1. A *P* corrisponde il codice della 'c' ovvero 3. Applicando la formula si ottiene C=73. Così per le altre due lettere 'i' e 'a' il codice crittografato varrà rispettivamente 59 e 1. Se si prova a decrittare si deve usare la formula 2. Per la lettera 'c' il cui codice crittografato vale 73 la potenza C^d vale 47477585226700098686074966922953 cifra più cifra meno (scherzo il calcolo è esatto) il cui modulo di n è proprio 73.

BLUM GOLDWASSER

Un altro interessante algoritmo di crittografia asimmetrica, per molti versi simile a RSA è quello di Blum Goldwasser, introdotto nel 1984. È classificato come algoritmo probabilistico semanticamente sicuro. La dimensione del cipher text aumenta di un modo lineare rispetto alla dimensione del plain text. Alla base dell'algoritmo c'è la generazione di una sequenza di numeri pseudo casuali attraverso l'algoritmo Blum Blum Shub (BBS). Come per RSA la sicurezza è legata alla "intrattabilità" del problema di fattorizzazione di numeri interi. Qualora si consideri un numero n come il prodotto *p*q* di due numeri primi molto grandi. Qui la sicurezza del metodo è basata soltanto sulla difficoltà insita nella fattorizzazione e non in altre specifiche come il residuo quadratico oppure assunzioni matematiche come per RSA. Anche la quantità di spazio occupata è un vantaggio visto che è proporzionale al plain text. Poiché per criptare si usa un algoritmo probabilistico uno stesso testo in chiaro può produrre in distinte applicazioni diversi testi criptati; è questa è un'ulteriore vantaggio del metodo. Purtroppo ha risultati pessimi ad attacchi conosciuti come "chosen cipher text" Entriamo nei particolari. Serviamoci sempre dei nostri amici Alice e Bob.

1. Generazione delle chiavi. Alice individua due numeri primi p e q molto grandi tali che entrambi siano congruenti a 3 modulo 4. Ciò significa che divisi per 4 danno resto 3. Successivamente, si calcola l'intero di Blum N=p*q. Così, la chiave pubblica è N mentre la chiave privata è la sua fattorizzazione p*q.
2. Criptazione del messaggio. Bob esprime il messaggio m come la sequenza di L bit (m0, m1, m2, … m_{L-1}). Poi sceglie un elemento casuale y compreso tra 1 e N e calcola $x_0 = y^2 \bmod N$ A questo punto Bob usa il generatore di numeri casuali come segue: con seme s vengono generati L bit casuali (b0, b1, b2, .. b_{L-1}) che saranno la chiave:

```
x0=s2
for i=0 to L-1 do
bi è il bit meno significativo di xi
Calcola xi=(xi-1)2 mod N
```

Poi Bob produce il cipher text come XOR tra i bit di m con quelli della chiave b.

```
c=b xor m
z=x0^(2^L) mod N
```

quindi invia la sequenza c (c0, c1, c2 … c_{L-1}) e z

3. Decrittografia
   Alice riceve (c0, c1, c2 … cL-1) e *z*. Può ripristinare *m* usando la procedura seguente:
   Mediante i fattori primi (p,q), Alice calcola $y_p = z^{((p+1)/4)^L} \bmod p$ e $y_p = z^{((q+1)/4)^L} \bmod q$
   Calcola il seme iniziale usato dal BBS

```
x0=q(q^-1 mod p)yp +p(p^-1 mod q) yp mod N
```

   Da x_0, calcola il vettore di bit *b* usando il generatore BBS, come nell'algoritmo di crittazione.
   Calcola il testo originale mediante XOR della chiave con il testo criptato: m=c xor b.
   Alice ha ottenuto così il messaggio *m*.

CONCLUSIONI

Continueremo ad esplorare il mondo della crittografia. In particolare nel prossimo numero analizzeremo ai raggi x un metodo che in questo periodo ha suscitato un particolare interesse, si tratta della crittografia quantistica.

Fabio Grimaldi

CHOSEN CHIPER TEXT

Si tratta di uno dei modelli d'attacco per criptare un messaggio cifrato. Il crittanalista sceglie un cipher text (da qui il nome del metodo) e prova a decriptare con chiavi non conosciute. Ovviamente, per portare a termine l'attacco è necessario del tempo (la quantità determina la bontà del metodo di crittografia o la fortuna dell'attaccante); normalmente tanto tempo. Spesso si usano delle macchine (computer) per tentare tutte le combinazioni possibili di chiavi nel minor tempo possibile. Per alcuni algoritmi questo attacco non sortisce nella stragrande maggioranza dei casi alcun risultato.

XOR

Si tratta di un operatore logico simile all'OR la cui tabella di verità è la seguente c=a xor b

a	b	c
0	0	0
0	1	1
1	0	1
1	1	0